

TortoiseSVN

Ein Subversion-Client für Windows

Version 1.6.16

**Stefan Küng
Lübbe Onken
Simon Large**

TortoiseSVN: Ein Subversion-Client für Windows: Version 1.6.16

von Stefan Küng, Lübbe Onken und Simon Large
Übersetzung: Stefan Küng, Lübbe Onken

Veröffentlicht 2011/01/21 21:21:17 (r20750)

Inhaltsverzeichnis

Vorwort	xi
1. Zielgruppe	xi
2. Lesetipps	xi
3. TortoiseSVN ist frei!	xii
4. Gemeinschaft	xii
5. Danksagung	xii
6. In diesem Dokument verwendete Terminologie	xii
1. Einleitung	1
1.1. Was ist TortoiseSVN?	1
1.2. Geschichte von TortoiseSVN	1
1.3. Eigenschaften von TortoiseSVN	1
1.4. Installation von TortoiseSVN	3
1.4.1. Systemanforderungen	3
1.4.2. Installation	3
1.4.3. Sprachpakete	3
1.4.4. Rechtschreibprüfung	3
2. Grundlagen der Versionskontrolle	5
2.1. Das Projektarchiv	5
2.2. Versions-Modelle	5
2.2.1. Das Problem des gemeinsamem Dateizugriffs	6
2.2.2. Die Sperren-Ändern-Freigeben Lösung	6
2.2.3. Die Kopieren-Ändern-Zusammenführen Lösung	7
2.2.4. Was macht Subversion?	9
2.3. Subversion bei der Arbeit	9
2.3.1. Arbeitskopien	9
2.3.2. Projektarchiv URLs	11
2.3.3. Revisionen	12
2.3.4. Wie Arbeitskopien das Projektarchiv verfolgen	13
2.4. Zusammenfassung	14
3. Das Projektarchiv	15
3.1. Projektarchiv erstellen	15
3.1.1. Ein Projektarchiv mit dem Kommandozeilen-Client erstellen	15
3.1.2. Erstellen eines Projektarchivs mit TortoiseSVN	15
3.1.3. Lokaler Zugriff auf das Projektarchiv	16
3.1.4. Projektarchiv auf einer Netzwerkfreigabe	16
3.1.5. Struktur des Projektarchivs	17
3.2. Projektarchiv sichern	18
3.3. Serverseitige Aktionsskripte	19
3.4. Auschecken aus Webseiten	19
3.5. Zugriff auf das Projektarchiv	20
3.6. Svnserve als Server	20
3.6.1. Einleitung	20
3.6.2. Svnserve installieren	21
3.6.3. Svnserve starten	21
3.6.4. Einfache Anmeldung mit Svnserve	23
3.6.5. Bessere Sicherheit mit SASL	24
3.6.6. Anmeldung mit svn+ssh	25
3.6.7. Pfadbasierte Autorisierung mit svnserve	25
3.7. Apache als Server	26
3.7.1. Einleitung	26
3.7.2. Apache installieren	26
3.7.3. Subversion installieren	27
3.7.4. Konfiguration	27
3.7.5. Mehrere Projektarchive	30
3.7.6. Pfadbasierte Autorisierung	30

3.7.7. Anmeldung an einer Windows Domäne	31
3.7.8. Mehrere Anmeldequellen	32
3.7.9. Den Server mit SSL sichern	33
3.7.10. Client-Zertifikate mit virtuellen SSL Hosts verwenden	35
4. Anleitung zum täglichen Gebrauch	37
4.1. Vorbereitungen	37
4.1.1. Überlagerte Symbole	37
4.1.2. Kontextmenüs	37
4.1.3. Ziehen und Ablegen	39
4.1.4. Tastaturkürzel	40
4.1.5. Anmeldung	40
4.1.6. Fenster maximieren	41
4.2. Daten in ein Projektarchiv importieren	41
4.2.1. Importieren	41
4.2.2. Import an Ort und Stelle	43
4.2.3. Spezielle Dateien	43
4.3. Eine Arbeitskopie auschecken	44
4.3.1. Rekursionstiefe	44
4.4. Ihre Änderungen ins Projektarchiv übertragen	46
4.4.1. Der Übertragen-Dialog	46
4.4.2. Änderungslisten	49
4.4.3. Objekte vom Übertragen ausschließen	49
4.4.4. Logmeldungen	49
4.4.5. Fortschrittsdialog	51
4.5. Aktualisieren der Arbeitskopie mit Änderungen von anderen	52
4.6. Konflikte auflösen	54
4.6.1. Dateikonflikte	54
4.6.2. Baumkonflikte	55
4.7. Statusinformationen anzeigen	58
4.7.1. Überlagerte Symbole	58
4.7.2. TortoiseSVN Spalten im Windows Explorer	60
4.7.3. Prüfe auf Änderungen	60
4.7.4. Unterschiede anzeigen	62
4.8. Änderungslisten	62
4.9. Log-Dialog	64
4.9.1. Den Log-Dialog starten	65
4.9.2. Aktionen im Revisionslog	65
4.9.3. Zusätzliche Informationen erhalten	66
4.9.4. Weitere Logmeldungen holen	70
4.9.5. Aktuelle Revision der Arbeitskopie	71
4.9.6. Datenintegration protokollieren	71
4.9.7. Ändern der Logmeldung und des Autors	72
4.9.8. Logmeldungen filtern	73
4.9.9. Statistiken anzeigen	74
4.9.10. Offline Modus	77
4.9.11. Die Ansicht aktualisieren	77
4.10. Unterschiede anzeigen	77
4.10.1. Datei-Unterschiede	78
4.10.2. Zeilenende- und Leerzeichenoptionen	79
4.10.3. Ordner vergleichen	79
4.10.4. Bilder mit TortoiseIDiff vergleichen	81
4.10.5. Externe Programme	82
4.11. Neue Dateien und Ordner hinzufügen	82
4.12. Dateien oder Ordner Kopieren/Umbenennen/Verschieben	83
4.13. Ignorieren von Dateien und Ordnern	84
4.13.1. Platzhalter in der Ignorieren-Liste	86
4.14. Löschen, Verschieben und Umbenennen	86
4.14.1. Löschen von Dateien und Ordnern	87

4.14.2. Dateien und Ordner verschieben	88
4.14.3. Groß-/Kleinschreibung eines Dateinamens ändern	89
4.14.4. Behandeln von Konflikten in der Groß-/Kleinschreibung	89
4.14.5. Externes Umbenennen reparieren	89
4.14.6. Nicht versionierte Dateien löschen	90
4.15. Änderungen rückgängig machen	90
4.16. Aufräumen	92
4.17. Projekt-Einstellungen	92
4.17.1. Subversion Eigenschaften	93
4.17.2. TortoiseSVN Projekteigenschaften	97
4.18. Externe Objekte	99
4.18.1. Externe Ordner	99
4.18.2. Externe Dateien	101
4.19. Verzweigen / Markieren	101
4.19.1. Einen Zweig oder eine Marke erstellen	102
4.19.2. Auschecken oder Wechseln...	103
4.20. Zusammenführen	104
4.20.1. Einen Revisionsbereich zusammenführen	106
4.20.2. Einen Zweig wieder eingliedern	107
4.20.3. Zusammenführen zweier Bäume	108
4.20.4. Optionen beim Zusammenführen	109
4.20.5. Ergebnisse des Zusammenführens betrachten	110
4.20.6. Verfolgung der Datenintegration	111
4.20.7. Behandeln von Konflikten beim Zusammenführen	111
4.20.8. Einen vollständigen Zweig zusammenführen	112
4.20.9. Wartung des Funktionszweiges	113
4.21. Sperren	113
4.21.1. Sperren von Dateien in Subversion	114
4.21.2. Eine Sperre erhalten	114
4.21.3. Eine Sperre freigeben	115
4.21.4. Den Sperrstatus prüfen	116
4.21.5. Nicht gesperrte Dateien mit Schreibschutz versehen	116
4.21.6. Aktionsskripte für Sperren	116
4.22. Erzeugen und Anwenden von Patches	117
4.22.1. Eine Patch-Datei erstellen	117
4.22.2. Eine Patchdatei anwenden	118
4.23. Wer hat welche Zeile geändert?	118
4.23.1. Annotieren für Dateien	119
4.23.2. Unterschiede annotieren	121
4.24. Projektarchivbetrachter	121
4.25. Revisionsgraphen	123
4.25.1. Knoten des Revisionsgraphen	124
4.25.2. Die Ansicht ändern	125
4.25.3. Den Graphen verwenden	127
4.25.4. Die Ansicht aktualisieren	128
4.25.5. Zweige ausdünnen	128
4.26. Eine Arbeitskopie exportieren	128
4.26.1. Eine Arbeitskopie aus der Versionskontrolle entfernen	130
4.27. Eine Arbeitskopie umplatzen	130
4.28. Integration mit einem System zur Fehlerverfolgung	131
4.28.1. Eintragsnummern in Logmeldungen einfügen	131
4.28.2. Informationen vom Fehlerverfolgungssystem beziehen	134
4.29. Integration mit webbasierten Projektarchivbetrachtern	135
4.30. TortoiseSVN Einstellungen	136
4.30.1. Allgemeine Einstellungen	136
4.30.2. Einstellungen des Revisionsgraphen	145
4.30.3. Überlagerte Symbole	147
4.30.4. Netzwerk Einstellungen	150

4.30.5. Einstellungen für externe Programme	152
4.30.6. Gespeicherte Daten	155
4.30.7. Log-Puffer	156
4.30.8. Clientseitige Aktionsskripte	159
4.30.9. TortoiseBlame Einstellungen	163
4.30.10. Einstellungen in der Registrierung	164
4.30.11. Subversion Arbeitsordner	165
4.31. Letzter Schritt	166
5. Das SubWCRev Programm	167
5.1. Die SubWCRev Kommandozeile	167
5.2. Schlüsselwortersetzung	168
5.3. Beispiele für Schlüsselwörter	168
5.4. COM Schnittstelle	169
6. IBugtraqProvider Schnittstelle	172
6.1. Die IBugtraqProvider Schnittstelle	172
6.2. Die IBugtraqProvider2 Schnittstelle	173
A. Häufig gestellte Fragen (FAQ)	177
B. Wie kann ich...	178
B.1. Viele Dateien auf einmal verschieben / kopieren	178
B.2. Anwender zwingen eine Logmeldung einzugeben	178
B.2.1. Aktionsskript auf dem Server	178
B.2.2. Projekteigenschaft setzen	179
B.3. Gezielt Dateien aus dem Projektarchiv aktualisieren	179
B.4. Revisionen im Projektarchiv rückgängig machen	179
B.4.1. Mit Hilfe des Log-Dialogs	179
B.4.2. Mit Hilfe des Zusammenführen-Dialogs	179
B.4.3. Mit Hilfe von svndumpfilter	180
B.5. Zwei Revisionen einer Datei oder eines Ordners vergleichen	180
B.6. Ein gemeinsames Unterprojekt einbinden	180
B.6.1. Die Eigenschaft svn:externals	181
B.6.2. Verschachtelte Arbeitskopien	181
B.6.3. Relative Pfade	181
B.7. Eine Verknüpfung zu einem Projektarchiv erstellen	182
B.8. Dateien ignorieren, die bereits unter Versionskontrolle sind	182
B.9. Eine Arbeitskopie aus der Versionskontrolle nehmen	182
B.10. Eine Arbeitskopie löschen	183
C. Tipps für Administratoren	184
C.1. TortoiseSVN über Gruppenrichtlinien verteilen	184
C.2. Die Versionsprüfung umleiten	184
C.3. Die SVN_ASP_DOT_NET_HACK Umgebungsvariable setzen	185
C.4. Kontextmenüeinträge deaktivieren	185
D. TortoiseSVN automatisieren	187
D.1. TortoiseSVN Befehle	187
D.2. TortoiseIDiff Befehle	190
E. Befehle der Kommandozeile	192
E.1. Grundregeln und Konventionen	192
E.2. TortoiseSVN Befehle	192
E.2.1. Auschecken	192
E.2.2. Aktualisieren	192
E.2.3. Aktualisieren zu Revision	193
E.2.4. Übertragen	193
E.2.5. Vergleich	193
E.2.6. Zeige Log	194
E.2.7. Prüfe auf Änderungen	194
E.2.8. Revisionsgraph	194
E.2.9. Projektarchivbetrachter	194
E.2.10. Konflikt bearbeiten	195
E.2.11. Konflikt aufgelöst	195

E.2.12. Umbenennen	195
E.2.13. Löschen	195
E.2.14. Rückgängig	195
E.2.15. Aufräumen	195
E.2.16. Sperre holen	195
E.2.17. Sperre freigeben	196
E.2.18. Verzweigen/Markieren	196
E.2.19. Wechseln zu	196
E.2.20. Zusammenführen	196
E.2.21. Exportieren	196
E.2.22. Umplatzieren	197
E.2.23. Projektarchiv hier erstellen	197
E.2.24. Hinzufügen	197
E.2.25. Importieren	197
E.2.26. Annotieren	197
E.2.27. Ignorieren	197
E.2.28. Erzeuge Patch	197
E.2.29. Patch anwenden	197
F. Implementierungsdetails	198
F.1. Überlagerte Symbole	198
G. Svnserve mit SSH sichern	200
G.1. Einen Linux Server einrichten	200
G.2. Einen Windows Server einrichten	200
G.3. SSH Werkzeuge zur Verwendung mit TortoiseSVN	201
G.4. Erzeugen von OpenSSH Zertifikaten	201
G.4.1. Schlüssel mittels ssh-keygen erzeugen	201
G.4.2. Schlüssel mittels PuTTYgen erzeugen	201
G.5. Test mittels PuTTY	201
G.6. SSH mit TortoiseSVN testen	202
G.7. SSH Konfigurationsvarianten	203
Glossar	204
Stichwortverzeichnis	208

Abbildungsverzeichnis

2.1. Ein typisches Client/Server System	5
2.2. Das zu vermeidende Problem	6
2.3. Die Sperren-Ändern-Freigeben Lösung	7
2.4. Die Kopieren-Ändern-Zusammenführen Lösung	8
2.5. ...Kopieren-Ändern-Zusammenführen fortgesetzt	8
2.6. Das Dateisystem des Projektarchivs	10
2.7. Das Projektarchiv	12
3.1. Das TortoiseSVN Menü für unversionierte Ordner	15
4.1. Explorer mit überlagerten Symbolen	37
4.2. Kontextmenü für einen Ordner unter Versionskontrolle	38
4.3. Explorer Kontextmenü für Verknüpfungen in einem versionierten Ordner	39
4.4. Rechts-Ziehen-Menü für einen Ordner unter Versionskontrolle	40
4.5. Anmeldedialog	41
4.6. Der Import-Dialog	42
4.7. Der Auschecken-Dialog	44
4.8. Der Übertragen-Dialog	47
4.9. Rechtschreibprüfung beim Eingeben einer Logmeldung	50
4.10. Eine laufende Übertragung im Fortschritts-Dialog	51
4.11. Der Fortschritts-Dialog nach Abschluss der Aktualisierung	52
4.12. Explorer mit überlagerten Symbolen	58
4.13. Prüfe auf Änderungen	60
4.14. Der Übertragen-Dialog mit Änderungsliste	63
4.15. Der Log-Dialog	65
4.16. Das Kontextmenü des Log-Dialogs	66
4.17. Kontextmenü des Log-Dialogs für zwei ausgewählte Revisionen	68
4.18. Kontextmenü der Dateiliste des Log-Dialogs	69
4.19. Der Log-Dialog mit bereits zusammengeführten Revisionen	72
4.20. Übertragungen per Autor als Histogramm	74
4.21. Übertragungen per Autor als Tortendiagramm	75
4.22. Übertragungen nach Datum	76
4.23. Offline gehen	77
4.24. Der Dialog zum Vergleichen von Revisionen	80
4.25. Ein Programm zum Vergleichen von Bildern	81
4.26. Explorer Kontextmenü für nicht versionierte Dateien	83
4.27. Rechts-Ziehen-Menü für einen Ordner unter Versionskontrolle	84
4.28. Explorer Kontextmenü für nicht versionierte Dateien	85
4.29. Explorer Kontextmenü für Dateien unter Versionskontrolle	87
4.30. Rückgängig-Dialog	91
4.31. Explorer Eigenschaftsseite, Subversion Tab	93
4.32. Subversion Eigenschaftsseite	94
4.33. Eigenschaften hinzufügen	95
4.34. Der Verzweigen/Markieren-Dialog	102
4.35. Der Wechseln-Zu-Dialog	104
4.36. Der Assistent - Revisionsbereich wählen	106
4.37. Der Assistent - Zweig wieder Eingliedern	108
4.38. Der Assistent - Zusammenführen von Bäumen	109
4.39. Der Rückfrage-Dialog für Konflikte	112
4.40. Der wieder Eingliedern Dialog	113
4.41. Der Sperren-Dialog	115
4.42. Der Dialog „Prüfe auf Änderungen“	116
4.43. Der „Erzeuge Patch“-Dialog.	117
4.44. Der Annotieren-Dialog	119
4.45. TortoiseBlame	120
4.46. Projektarchivbetrachter	122
4.47. Ein Revisionsgraph	124

4.48. Exportiere von URL	129
4.49. Der Umplatzieren-Dialog	130
4.50. Beispielabfrage des Fehlerverfolgungssystems	135
4.51. Der Einstellungsdialog, Allgemein	137
4.52. Der Einstellungsdialog, Kontextmenü	139
4.53. Einstellungen Dialoge, Seite 1	140
4.54. Einstellungen Dialoge, Seite 2	142
4.55. Der Einstellungsdialog, Farben	143
4.56. Der Einstellungsdialog, Revisionsgraph	145
4.57. Der Einstellungsdialog, Farben des Revisionsgraphen	146
4.58. Der Einstellungsdialog, Symbolauswahl	147
4.59. Der Einstellungsdialog, Symbolauswahl	150
4.60. Der Einstellungsdialog, Netzwerkseite	150
4.61. Der Einstellungsdialog, Externe Programme	152
4.62. Erweiterte Einstellungen für Vergleichs- und Konflikteditor	154
4.63. Der Einstellungsdialog, gespeicherte Daten	155
4.64. Der Einstellungsdialog, Log-Puffer	156
4.65. Der Einstellungsdialog, Log-Puffer Statistiken	158
4.66. Der Einstellungsdialog, Aktionsskripte	159
4.67. Der Einstellungsdialog, Aktionsskripte einrichten	160
4.68. Der Einstellungsdialog, Integration eines Fehlerverfolgungssystems	162
4.69. Der Einstellungsdialog, TortoiseBlame	163
C.1. Die Versionsprüfung	184

Tabellenverzeichnis

2.1. Projektarchiv Zugriffs-URLs	11
3.1. Apache <code>httpd.conf</code> Einstellungen	28
5.1. Liste der Kommandozeilenoptionen	167
5.2. Liste der Kommandozeilenoptionen	168
5.3. Unterstützte COM-Automatisierungen	169
C.1. Menüeinträge und ihre Werte	185
D.1. Liste der Befehle und Parameter	188
D.2. Liste der Parameter	190

Vorwort



TortoiseSVN

- Arbeiten Sie in einem Team?
- Ist es Ihnen jemals passiert, dass Sie an einer Datei Änderungen vorgenommen haben und jemand anders dieselbe Datei verändert hat? Haben Sie dadurch Ihre eigenen Änderungen oder die des anderen verloren?
- Haben Sie jemals eine Datei abgespeichert und dann später Ihre Änderungen wieder rückgängig machen wollen? Haben Sie sich jemals gewünscht, Sie könnten nachsehen, wie eine Datei zu einem früheren Zeitpunkt ausgesehen hat?
- Haben Sie jemals einen Fehler in Ihrem Projekt gefunden und wollten wissen, wann und durch wen dieser Fehler in das Projekt geraten ist?

Falls Sie eine der obigen Fragen mit „Ja“ beantwortet haben, dann ist TortoiseSVN das Richtige für Sie! Lesen Sie weiter und finden Sie heraus, wie TortoiseSVN Ihnen bei Ihrer Arbeit helfen kann. Es ist nicht schwierig.

1. Zielgruppe

Dieses Buch ist für Computerbenutzer geschrieben, welche Subversion für die Handhabung ihrer Daten nutzen wollen, aber den Umgang mit der Kommandozeile nicht gewohnt sind. Da TortoiseSVN eine Windows Shell-Erweiterung ist, wird angenommen, dass der Leser den Umgang mit dem Explorer gewohnt ist und weiß, wie man diesen benutzt.

2. Lesetipps

Dieses **Vorwort** erklärt das TortoiseSVN Projekt, die Gemeinschaft von Leuten die daran arbeiten sowie die Lizenzbedingungen und die Bedingungen für die Weitergabe der Software.

In **Kapitel 1, Einleitung** wird erklärt, was TortoiseSVN ist, wie es funktioniert, woher es kommt und wie es auf Ihrem PC installiert wird.

In **Kapitel 2, Grundlagen der Versionskontrolle** geben wir eine kurze Einführung in das *Subversion* Versionskontrollsystem, welches TortoiseSVN zugrunde liegt. Dieses Kapitel stammt aus der Dokumentation des Subversion Projekts und erklärt die verschiedenen Ansätze zur Versionskontrolle und wie Subversion arbeitet.

In **Kapitel 3, Das Projektarchiv** wird erklärt, wie ein lokales Projektarchiv angelegt wird, damit Sie TortoiseSVN und Subversion auf einem einzigen PC testen können. Weiterhin wird auf die Verwaltung von Projektarchiven eingegangen, was für Projektarchive, die sich auf einem Server befinden, relevant ist. Darüber hinaus gibt es einen Abschnitt, der sich mit dem Einrichten eines Subversion Servers befasst.

Kapitel 4, Anleitung zum täglichen Gebrauch ist das wichtigste Kapitel der Dokumentation, da darin alle Funktionen von TortoiseSVN sowie ihre Verwendung erklärt werden. Es hat die Form einer Anleitung, beginnend mit dem Auschecken einer Arbeitskopie, Durchführen von Änderungen, Übertragen der Änderungen und so weiter. In den weiteren Abschnitten wird auf fortgeschrittene Themen eingegangen.

Kapitel 5, Das SubWCRev Programm ist ein separates Programm des TortoiseSVN Pakets, das Informationen aus einer Arbeitskopie extrahieren und in eine Datei schreiben kann. Dieses Programm ist nützlich, um Informationen beim Erzeugen in Ihr Projekt einzubinden.

Der **Anhang B, *Wie kann ich...*** beantwortet einige allgemeine Fragen zu Aufgaben, die anderweitig nicht beschrieben werden.

Der **Anhang D, *TortoiseSVN automatisieren*** beschreibt, wie die TortoiseSVN GUI-Dialoge von der Kommandozeile aus aufgerufen werden können. Dies ist nützlich, wenn Sie TortoiseSVN aus Skripten heraus aufrufen aber trotzdem noch Benutzereingaben benötigen.

In **Anhang E, *Befehle der Kommandozeile*** geben einen Überblick über die Zusammenhänge zwischen TortoiseSVN Befehlen und ihren Äquivalenten in dem Subversion Kommandozeilenprogramm `svn.exe`.

3. TortoiseSVN ist frei!

TortoiseSVN ist freie Software. Sie brauchen nichts dafür zu bezahlen. Sie können das Programm benutzen wie Sie wollen. TortoiseSVN wird unter der GNU „General Public License (GPL)“ entwickelt.

TortoiseSVN ist ein Open Source Projekt. Das bedeutet, dass sie Zugang zum Quellcode des Programms haben. Sie können den Quellcode unter folgender URL <http://code.google.com/p/tortoisesvn/source/browse/> ansehen. Sie werden nach einem Benutzernamen und Passwort gefragt. Der Benutzername ist `guest` und das Passwort muss leer gelassen werden. Die aktuellste Version (an der wir gerade arbeiten) ist immer im Ordner `/trunk/` zu finden. Die bereits veröffentlichten Versionen finden sich im Ordner `/tags/`.

4. Gemeinschaft

Sowohl TortoiseSVN als auch Subversion werden von verschiedenen Gruppen von Leuten entwickelt. Diese Leute kommen aus vielen verschiedenen Ländern der Welt und arbeiten zusammen, um diese tollen Programme weiter zu entwickeln.

5. Danksagung

Tim Kemp
für die Gründung des TortoiseSVN Projekts

Stefan Küng
für die harte Arbeit an TortoiseSVN und die Programmierarbeit

Lübbe Onken
für die schönen Symbole, das Logo, die Dokumentation und das Übersetzungsmanagement

Simon Large
für die Hilfe bei der Dokumentation und bei der Fehlerjagd

Das Subversion Buch
für die großartige Einführung in Subversion und das Kapitel 2, das wir hier kopiert haben.

Das Tigris Style Projekt
für einige der Stile, welche für diese Dokumentation verwendet wurden.

Unsere Helfer
für die Patches, Fehlermeldungen und dafür, dass sie anderen helfen, indem Sie Fragen auf unserer Mailingliste beantworten

Unsere Spender
für viele Stunden voll toller Musik, welche sie uns geschenkt haben.

6. In diesem Dokument verwendete Terminologie

Um das Lesen der Dokumentation einfacher zu gestalten, sind die Namen aller Masken und Menüs von TortoiseSVN in einer anderen Schrift hervorgehoben. Der Log-Dialog zum Beispiel.

Eine Menüauswahl wird durch einen Pfeil angedeutet. TortoiseSVN → Zeige Log bedeutet: Wählen Sie *Zeige Log* aus dem *TortoiseSVN* Kontextmenü.

Ein lokales Kontextmenü in einem TortoiseSVN Dialog wird folgendermaßen angezeigt: Kontextmenü → Speichern Unter...

Schaltflächen werden durch einen 3D Effekt hervorgehoben: Drücken Sie OK, um weiterzumachen.

Benutzeraktionen werden durch Fettdruck hervorgehoben. **Alt+A**: Drücken Sie die **Alt**-Taste und während Sie diese gedrückt halten, die **A**-Taste dazu. Rechts-Ziehen: Halten Sie die rechte Maustaste gedrückt und *ziehen* Sie die gewählten Einträge an den neuen Ort.

Systemausgaben und Tastatureingaben werden durch einen anderen Zeichensatz hervorgehoben.



Wichtig

Wichtige Hinweise im Dokument.



Tipp

Tipps, die Ihre Arbeit mit Subversion erleichtern.



Achtung

Stellen, an denen Sie vorsichtig sein müssen, was sie tun.



Warnung

Stellen, an denen extreme Vorsicht geboten ist. Nichtbeachtung dieser Warnungen kann zu irreparablen Datenverlusten führen.



Kapitel 1. Einleitung

Versionskontrolle ist die Kunst, die Kontrolle über Änderungen an Informationen zu behalten. Es ist seit langer Zeit ein unverzichtbares Hilfsmittel für Programmierer, welche ihre Zeit damit verbringen, jeweils kleine Änderungen an Software vorzunehmen und dann manchmal diese Änderungen am nächsten Tag zurücknehmen. Stellen Sie sich ein Team von solchen Programmierern vor, welche gleichzeitig an demselben Projekt - und dabei unter Umständen auch noch gleichzeitig an denselben Dateien! - arbeiten und Sie können sehen, warum ein gutes Versionskontrollsystem notwendig ist, um ein mögliches Chaos zu verhindern.

1.1. Was ist TortoiseSVN?

TortoiseSVN ist ein freies Open-Source Programm für das Subversion Versionskontrollsystem. Das heißt, TortoiseSVN verwaltet Dateien und Ordner über die Zeit. Dateien werden in einem zentralen *Projektarchiv* gespeichert. Das Projektarchiv entspricht einem normalen Dateiserver, mit der Besonderheit, dass sämtliche Änderungen protokolliert werden. Dies erlaubt es, ältere Versionen von Dateien zurückzuholen und Änderungen über die Zeit zu verfolgen. Aus diesem Grund sehen viele Leute Subversion als eine Art „Zeitmaschine“ an.

Einige Versionskontrollsysteme sind zugleich Programmkonfigurations-Managementsysteme (Software Configuration Management Systems, SCM). Diese Systeme sind speziell dazu ausgelegt, Quellcode zu handhaben und haben viele Funktionen, welche spezifisch für die Softwareentwicklung sind - so verstehen diese etwa verschiedene Programmiersprachen oder stellen Hilfsmittel für die Erstellung von Programmen zur Verfügung. Subversion hingegen ist kein solches System; es ist allgemein gehalten, so dass es für die Handhabung von *beliebigen* Dateien, einschließlich Quellcode, geeignet ist.

1.2. Geschichte von TortoiseSVN

Tim Kemp fand im Jahr 2002, dass Subversion ein sehr gutes Versionskontrollsystem sei, aber es fehle eine gute grafische Oberfläche dazu. Die Idee einer Windows Shell-Integration wurde von dem Windows-Programm TortoiseCVS für das ältere CVS Versionskontrollsystem inspiriert.

Tim studierte den Quellcode von TortoiseCVS und benutzte diesen als Basis für TortoiseSVN. Er startete dann das Projekt, registrierte die Domäne `tortoisesvn.org` und stellte den Quellcode online zur Verfügung. Während dieser Zeit suchte Stefan Küng nach einem guten Versionskontrollsystem und fand Subversion und den Quellcode von TortoiseSVN. Da TortoiseSVN noch nicht für den normalen Gebrauch nutzbar war, fing er an, TortoiseSVN weiter zu entwickeln. Schon bald hatte er den größten Teil des Programmcodes neu geschrieben und fügte Funktionen und Befehle hinzu bis zu einem Punkt, wo von dem ursprünglichen Code nichts mehr übrig war.

Während Subversion immer stabiler wurde, zog es immer mehr Benutzer an, welche auch TortoiseSVN als ihren Subversion Client verwenden. Die Benutzerzahl wuchs schnell (und wächst täglich weiter). Zu dieser Zeit bot Lübke Onken an, beim Design zu helfen und kreierte die Symbole und das Logo für TortoiseSVN. Er kümmert sich seitdem auch um die Webseite und das Übersetzungsmanagement.

1.3. Eigenschaften von TortoiseSVN

Was macht TortoiseSVN zu einem guten Subversion Client? Hier eine Liste der herausragenden Eigenschaften:

Shell Integration

TortoiseSVN fügt sich nahtlos in die Windows Shell (den Explorer) ein. Das heißt, Sie können weiter mit den gewohnten Programmen arbeiten und müssen sich nicht an ein neues Programm gewöhnen. Und brauchen auch nicht jedes mal erst in ein anderes Programm zu wechseln, wenn Sie Funktionen der Versionskontrolle benötigen!

Und dabei sind Sie nicht einmal gezwungen, den Windows Explorer zu benutzen. Auch in anderen Dateimanagern stehen Ihnen prinzipiell das Kontextmenü und die anderen Funktionen von TortoiseSVN jederzeit zur Verfügung. Wobei allerdings zu beachten ist, dass TortoiseSVN speziell als Erweiterung des Windows Explorers entwickelt wird. Daher kann es sein, dass in anderen Programmen die Integration nicht vollständig ist und beispielsweise die überlagerten Symbole nicht angezeigt werden.

Überlagerte Symbole

Der Status von jeder Datei unter Versionskontrolle wird durch ein kleines überlagertes Symbol angezeigt. Auf diese Weise können Sie sofort den Zustand Ihrer Arbeitskopie erkennen.

Einfacher Zugriff auf Subversion-Befehle

Alle Befehle von Subversion sind über das Kontextmenü des Explorers zugänglich; TortoiseSVN fügt dort ein eigenes Untermenü ein.

Da TortoiseSVN ein Subversion Client ist, möchten wir auch ein paar Features von Subversion selbst erwähnen:

Versionierung von Verzeichnissen

CVS speichert nur die Geschichte von einzelnen Dateien, aber Subversion implementiert ein „virtuelles“ kontrolliertes Dateisystem, welches auch Änderungen an Ordnern über die Zeit speichert. Dateien *und* Ordner werden kontrolliert. Daher gibt es Befehle, um Dateien/Ordner zu **verschieben**. Und der **Kopiere...** Befehl arbeitet mit Dateien *und* Ordnern.

Atomare Übertragungen

Eine Übertragung von Änderungen geht entweder komplett in das Projektarchiv oder gar nicht. Das erlaubt Entwicklern, Änderungen als zusammenhängende logische Einheit zu übertragen.

Versionierte Metadaten

Jede Datei und jeder Ordner hat einen unsichtbaren Satz von „Eigenschaften“ angehängt. Sie können selbst neue Eigenschaften definieren und jede Art von Schlüssel/Wertepaare speichern. Änderungen an Eigenschaften werden versioniert, genauso wie der normale Dateiinhalt.

Auswahl an Netzwerkschichten

Subversion hat einen abstrakten Begriff des Zugriffs auf das Projektarchiv, der es einfach macht, neue Netzwerkschichten zu implementieren. Subversions am weitesten fortgeschrittener Netzwerkschicht ist ein Modul für den Apache Web-Server, welcher eine Variante von HTTP genannt WebDAV/DeltaV nutzt. Dies gibt Subversion einen Vorteil an Stabilität und Interoperabilität, und stellt verschiedene Schlüsselfunktionen gratis zur Verfügung: Anmeldung, Autorisierung, Netzwerkdatenkompression, Verschlüsselung via SSL und das Betrachten des Projektarchivs in einem Web-Browser zum Beispiel. Ein kleinerer Subversion Server ist ebenfalls vorhanden. Dieser Server nutzt ein eigenes Netzwerkprotokoll, welches über SSH getunnelt werden kann.

Konsistente Datenhaltung

Subversion drückt Dateiunterschiede mit Hilfe eines binären Differenzalgorithmus aus, welcher identisch für Text- und Binärdateien arbeitet. Beide Dateiarten werden gleich komprimiert im Projektarchiv abgespeichert, und jeweils nur diese Differenz wird in beiden Richtungen übers Netzwerk übertragen.

Effizientes Verzweigen und Markieren

Die Kosten für Verzweigen und Markieren müssen nicht proportional zur Projektgröße sein. Subversion erstellt Zweige und Marken durch einfaches Kopieren des Projektes. Dabei wird ein Mechanismus ähnlich dem eines Verweises verwendet. Diese Operationen benötigen deshalb nur wenig Zeit und Speicherplatz.

Wartbarkeit

Subversion hat keine Altlasten; es ist als Sammlung von C-Bibliotheken mit gut definierten APIs implementiert. Dies macht Subversion extrem gut wartbar und durch andere Programme und Programmiersprachen nutzbar.

1.4. Installation von TortoiseSVN

1.4.1. Systemanforderungen

TortoiseSVN läuft auf Windows 2000 SP2, WinXP oder höher. Windows 98, Windows ME und Windows NT4 werden seit TortoiseSVN 1.2.0 nicht mehr unterstützt, aber Sie können die älteren Installer noch herunterladen, falls Sie diese benötigen.

Falls Sie irgendwelche Probleme während oder nach der Installation von TortoiseSVN haben sollten, sehen Sie bitte zunächst im Kapitel [Anhang A, Häufig gestellte Fragen \(FAQ\)](#) nach.

1.4.2. Installation

TortoiseSVN wird als selbst installierende Anwendung ausgeliefert. Starten Sie die Installation durch einen Doppelklick auf die Datei und folgen Sie den Anweisungen. Das Installationsprogramm wird sich um alles weiter kümmern.



Wichtig

Sie benötigen Administratorrechte, um TortoiseSVN zu installieren

1.4.3. Sprachpakete

Die TortoiseSVN Benutzerschnittstelle wurde in viele verschiedene Sprachen übersetzt, so dass auch für Sie ein passendes Sprachpaket zur Verfügung stehen könnte. Sie finden die vorhandenen Sprachpakete auf unserer [Übersetzungsseite](http://tortoisesvn.net/translation_status) [http://tortoisesvn.net/translation_status]. Wenn für Sie nichts Passendes dabei ist, werden Sie doch einfach Projektmitglied und steuern Sie Ihre eigene Übersetzung bei ;-)

Jedes Sprachpaket ist in einem .exe Installer enthalten. Lassen Sie das Installationsprogramm einmal laufen. Nach dem nächsten Neustart Ihres PC, steht die Übersetzung zur Verfügung.

1.4.4. Rechtschreibprüfung

TortoiseSVN enthält eine Rechtschreibprüfung, die es Ihnen ermöglicht, die Logmeldungen zu prüfen. Dies ist besonders dann nützlich, wenn die Projektsprache nicht Ihre Muttersprache ist. Die Rechtschreibprüfung verwendet die gleichen Wörterbücher wie [OpenOffice](http://openoffice.org) [http://openoffice.org] und [Mozilla](http://mozilla.org) [http://mozilla.org].

Das Installationsprogramm bringt die englische und amerikanische Rechtschreibprüfung mit. Wenn Sie weitere Sprachen benötigen, installieren Sie einfach eines von TortoiseSVN's Sprachpaketen. Diese bringen außer der lokalisierten Anwenderoberfläche auch die entsprechende Rechtschreibprüfung mit. Beim nächsten Start wird das Wörterbuch dann zur Verfügung stehen.

Sie können die Wörterbücher auch selbst installieren. Wenn Sie bereits OpenOffice oder Mozilla auf Ihrem PC haben, können Sie deren Wörterbücher aus dem Anwendungsverzeichnis kopieren. Andernfalls müssen Sie die Wörterbücher von <http://wiki.services.openoffice.org/wiki/Dictionaries> herunterladen.

Wenn Sie die Wörterbücher haben, müssen Sie diese möglicherweise erst umbenennen so dass die Dateinamen nur die Länderabkürzungen aufweisen. Zum Beispiel:

- en_US.aff
- en_US.dic

Danach kopieren Sie diese Dateien einfach im Installationsverzeichnis von TortoiseSVN in das Unterverzeichnis Languages, das Sie eventuell vorher noch anlegen müssen C:

\Programme\TortoiseSVN\Languages. Wenn Sie TortoiseSVN das nächste Mal starten, ist die Rechtschreibkorrektur bereit.

Falls Sie mehrere Wörterbücher installieren, verwendet TortoiseSVN die folgenden Regeln, um das zu verwendende Wörterbuch zu ermitteln.

1. Prüfe die `tsvn:projectlanguage` Einstellung. Lesen Sie in [Abschnitt 4.17](#), „Projekt-Einstellungen“ nach, wie Sie Projekteinstellungen vornehmen.
2. Wenn keine Projektsprache eingestellt ist oder das entsprechende Sprachpaket nicht installiert ist, versuche ein zur Windows Spracheinstellung passendes Paket zu finden.
3. Wenn kein Paket mit der Windows Spracheinstellung übereinstimmt, versuche es mit der „Basis“-Sprache, z.B. `de_CH` (Schweizerdeutsch) fällt zurück auf `de_DE` (Deutsch).
4. Wenn keines der obenstehenden Verfahren funktioniert, wird Englisch verwendet, das in der Standardinstallation enthalten ist.

Kapitel 2. Grundlagen der Versionskontrolle

Dieses Kapitel ist eine leicht veränderte Version desselben Kapitels aus dem Subversion Buch. Das vollständige Subversion Buch findet Sie unter folgender Adresse: <http://svnbook.red-bean.com/> [<http://svnbook.red-bean.com/>].

Dieses Kapitel ist eine kurze, beiläufige Einführung in Subversion. Falls Sie noch nie etwas von Versionskontrolle gehört haben, dann ist dieses Kapitel definitiv etwas für Sie. Wir beginnen mit einer Diskussion über allgemeine Konzepte der Versionskontrolle, arbeiten uns dann vor in die spezifischen Ideen hinter Subversion und zeigen ein paar einfache Beispiele zur Benutzung von Subversion.

Obwohl die hier gezeigten Beispiele Personen zeigen, die mit Quellcode arbeiten, beachten Sie bitte, dass Subversion *jedliche* Art von Daten verarbeiten kann - es ist nicht auf Quellcode beschränkt.

2.1. Das Projektarchiv

Subversion ist ein zentralisiertes System für die Verteilung von Informationen. In seinem Kern befindet sich eine Datenbank, welche die Informationen in Form eines Dateibaumes - einer typischen Hierarchie von Dateien und Ordnern - abspeichert. Eine beliebige Anzahl von Clients kontaktiert das Projektarchiv und liest und schreibt von/in diese Dateien. Durch ein Schreiben von Daten stellt ein Client diese Daten auch anderen zur Verfügung, durch Lesen von Daten bekommt ein Client die Informationen von anderen.

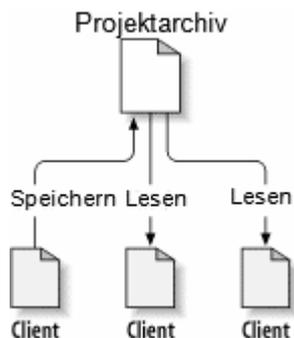


Abbildung 2.1. Ein typisches Client/Server System

Warum ist dies so interessant? Bis jetzt klingt dies alles wie die Definition eines typischen Dateiservers. Und tatsächlich *ist* das Projektarchiv eine Art Dateiserver, aber sie ist kein typischer Dateiserver. Was das Subversion Projektarchiv so speziell macht, ist, dass diese *jede je gemachte Änderung speichert*: jede Änderung an einer Datei, sogar Änderungen am Dateibaum selbst wie das Hinzufügen, Löschen oder Umbenennen/Verschieben von Dateien und Ordnern.

Wenn ein Client Daten aus dem Projektarchiv liest, so sieht er normalerweise nur die aktuellste Version des Dateibaumes. Aber ein Clientprogramm hat auch die Möglichkeit, *frühere* Versionen zu sehen. Zum Beispiel kann ein Client Fragen stellen wie „Was enthielt dieser Ordner letzten Mittwoch?“ oder „Wer war die letzte Person, welche diese Datei geändert hat und was für Änderungen waren dies?“. Dies sind die Art von Fragen, die das Herz eines Versionskontrollsystems bilden: Systeme, welche gebaut wurden um Änderungen aufzuzeichnen und über die Zeit hinweg zu verfolgen.

2.2. Versions-Modelle

Alle Versionskontrollsysteme haben ein gemeinsames, grundlegendes Problem zu lösen: Wie kann das System verschiedenen Benutzern erlauben, die Informationen zu teilen, aber gleichzeitig verhindern,

dass die Benutzer sich gegenseitig auf die Füße treten? Es ist viel zu einfach für Benutzer, ungewollt Änderungen von anderen Benutzern zu überschreiben.

2.2.1. Das Problem des gemeinsamen Dateizugriffs

Stellen Sie sich folgende Szene vor: angenommen, wir haben zwei Mitarbeiter, Harry und Sally. Sie beide beschließen, dieselbe Datei gleichzeitig zu ändern. Wenn Harry seine Änderungen zuerst im Projektarchiv speichert, dann ist es möglich (ein paar Augenblicke später), dass Sally ungewollt diese Änderungen durch ihre eigenen überschreibt. Obwohl die Änderungen von Harry nicht verloren sind (das Versionskontrollsystem speichert *alle* Änderungen), werden Harrys Änderungen in der aktuellsten Version von Sally nicht vorhanden sein, weil Sally diese Änderungen nie gesehen hat. Harrys Änderungen sind also trotzdem verloren - oder zumindest fehlen diese in der aktuellsten Version. Dies ist definitiv eine Situation, die wir verhindern wollen!

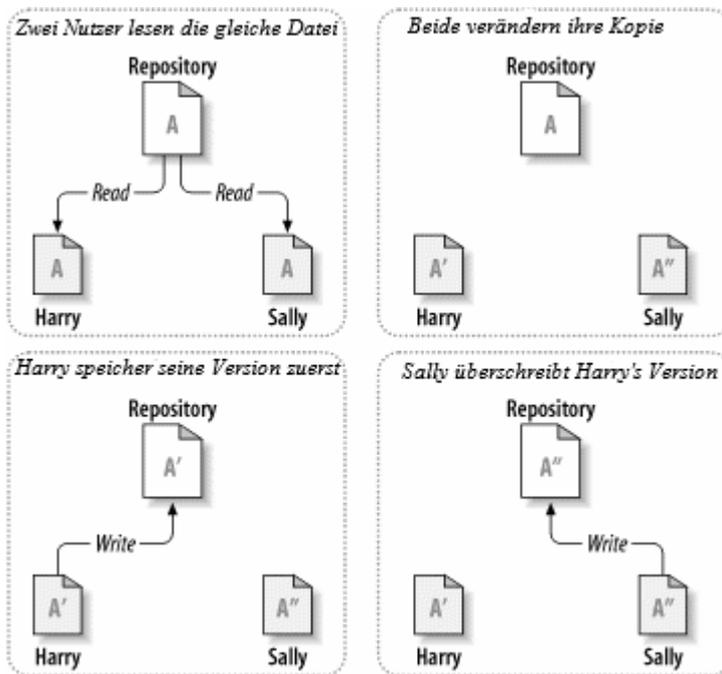


Abbildung 2.2. Das zu vermeidende Problem

2.2.2. Die Sperren-Ändern-Freigeben Lösung

Viele Versionskontrollsysteme nutzen ein Sperren-Ändern-Freigeben Modell, was eine sehr einfache Lösung ist. In einem solchen System kann jeweils nur eine einzige Person eine Datei ändern. Zuerst muss Harry die Datei *sperren*, bevor er mit dem Ändern beginnen kann. Sperren ist so ähnlich wie das Ausleihen eines Buchs aus der Bibliothek; wenn Harry die Datei gesperrt hat, dann kann Sally keine Änderungen an dieser Datei mehr vornehmen. Wenn sie es dennoch versucht, kriegt sie eine Fehlermeldung. Alles, was sie tun kann, ist die Datei lesen und zu warten, bis Harry die Datei wieder freigegeben hat (das Buch in die Bibliothek zurückgebracht hat). Danach kann Sally die Datei sperren und ihre Änderungen vornehmen.

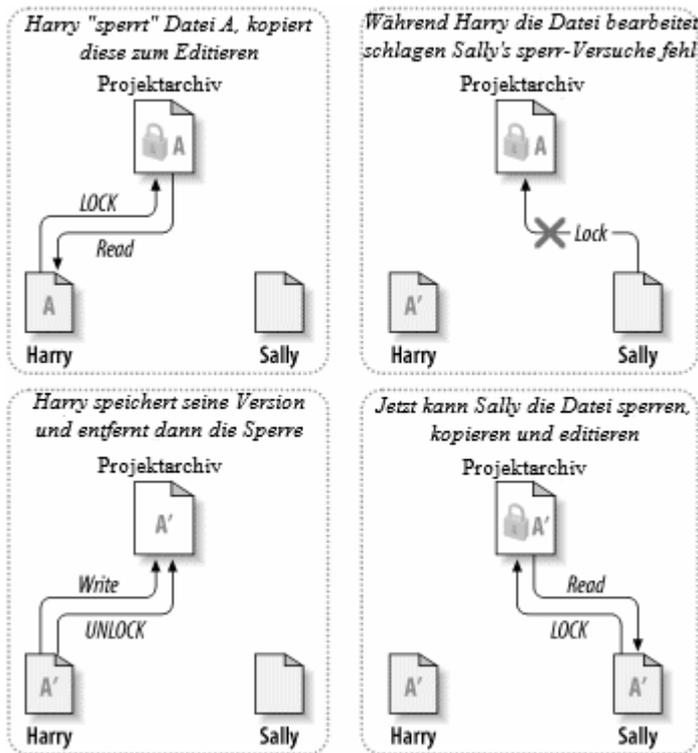


Abbildung 2.3. Die Sperren-Ändern-Freigeben Lösung

Das Problem mit der Sperren-Ändern-Freigeben Lösung ist, dass diese zu restriktiv ist und oft zu einem Hindernis für die Benutzer wird:

- *Sperren kann administrative Probleme auslösen.* Manchmal wird Harry eine Datei sperren und dann vergessen, diese wieder freizugeben. In der Zwischenzeit wartet Sally bis sie diese Datei ändern kann, aber durch die Sperre von Harry sind ihr die Hände gebunden. Und dann geht Harry in die Ferien. Nun braucht Sally einen Administrator, um die Sperre von Harry aufzuheben. Diese Situation führt zu unnötigen Verzögerungen und verschwendet Zeit.
- *Sperren kann zu unnötiger sequenzieller Arbeitsweise führen.* Was, wenn Harry den Anfang einer Textdatei ändert und Sally das Ende dieser Textdatei ändern möchte? Diese Änderungen stören sich überhaupt nicht und könnten sehr einfach gleichzeitig ausgeführt werden, in der Annahme, dass diese Änderungen ohne Probleme zusammengeführt werden. Es ist in dieser Situation völlig unnötig zu warten, bis der eine mit der Arbeit fertig ist.
- *Sperren kann ein falsches Gefühl von Sicherheit erzeugen.* Angenommen, Harry sperrt und ändert eine Datei A, während Sally eine Datei B gesperrt hat und ändert. Aber angenommen, diese beiden Dateien sind nicht unabhängig voneinander und die Änderungen an der einen Datei sind nicht kompatibel mit den Änderungen an der anderen Datei. Plötzlich funktionieren die Dateien A und B nicht mehr zusammen. Das Sperr-System konnte dieses Problem nicht verhindern - es spiegelte sogar eine falsche Sicherheit vor. Es ist einfach für Harry und Sally anzunehmen, dass durch das Sperren von Dateien jeder Änderungen sicher durchführen kann.

2.2.3. Die Kopieren-Ändern-Zusammenführen Lösung

Subversion, CVS und andere Versionskontrollsysteme nutzen ein Kopieren-Ändern-Zusammenführen Modell als Alternative zur Sperrlösung. In diesem Modell hat jeder Benutzer eine persönliche *Arbeitskopie* des Projekts. Die Benutzer arbeiten dann parallel und ändern jeweils ihre persönliche Arbeitskopie. Schließlich werden diese privaten Arbeitskopien zu einer endgültigen Version zusammengeführt. Das Versionskontrollsystem hilft beim Zusammenführen, überlässt die endgültige Kontrolle aber dem Benutzer selbst.

Hier ein Beispiel. Angenommen, Harry und Sally haben jeder eine private Arbeitskopie von demselben Projekt. Sie arbeiten parallel und ändern dieselbe Datei A in ihren Arbeitskopien. Sally speichert ihre Änderungen zuerst im Projektarchiv. Wenn Harry nun dasselbe versucht, bekommt er eine Meldung vom Projektarchiv, dass seine Datei nicht aktuell ist. In anderen Worten, die Datei A wurde irgendwie im Projektarchiv verändert, seit er sie das letzte Mal aktualisiert hat. Also *aktualisiert* Harry seine Arbeitskopie mit dem Projektarchiv und *führt* die Änderungen im Projektarchiv mit seinen eigenen zusammen. Die Wahrscheinlichkeit, dass die Änderungen von Sally sich nicht mit seinen eigenen beißen, ist groß, und so kann Harry gleich danach die Datei im Projektarchiv speichern.

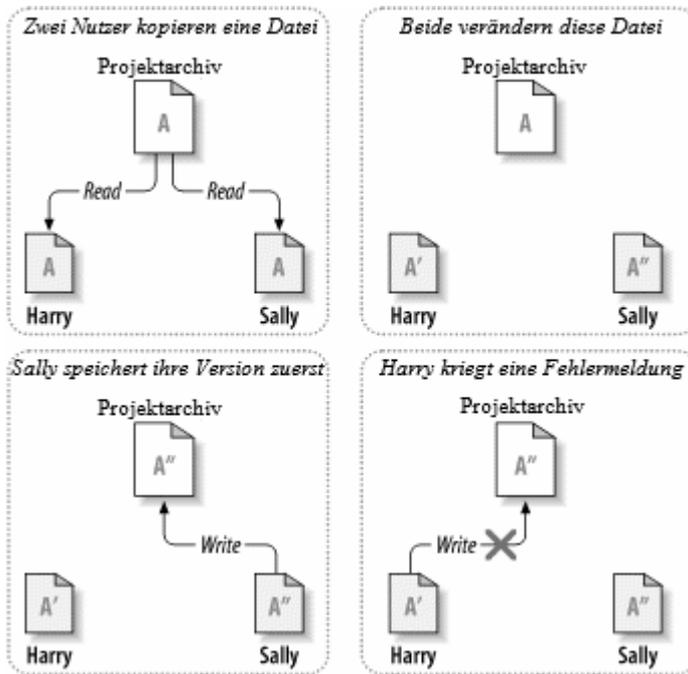


Abbildung 2.4. Die Kopieren-Ändern-Zusammenführen Lösung

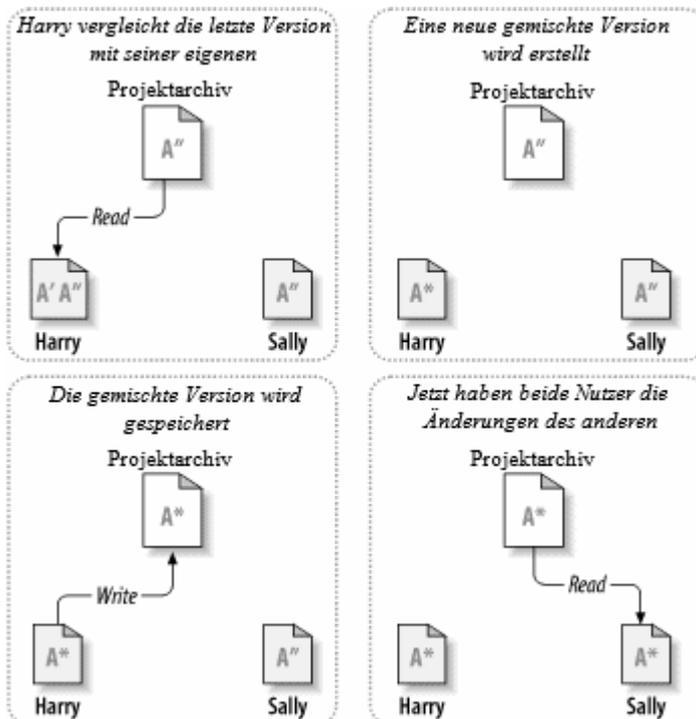


Abbildung 2.5. ...Kopieren-Ändern-Zusammenführen fortgesetzt

Aber was, wenn die Änderungen von Sally sich mit Harrys Änderungen *tatsächlich* überlappen? Was dann? Diese Situation wird ein *Konflikt* genannt und ist üblicherweise kein großes Problem. Wenn Harry in diesem Falle seine Arbeitskopie aktualisiert und die Änderungen von Sally mit den seinen zusammenführt, wird seine Kopie der Datei A als „In Konflikt“ markiert; er kann dann beide Versionen der Datei ansehen und manuell zwischen den Änderungen von Sally oder seinen eigenen wählen. Beachten Sie, dass Software solche Konflikte nicht selbst lösen kann; nur Menschen sind in der Lage, die Änderungen auch zu verstehen und die notwendigen intelligenten Entscheidungen (vielleicht durch ein Gespräch mit Sally) zu treffen.

Das Kopieren-Ändern-Zusammenführen Modell mag ein wenig chaotisch aussehen, aber in der Praxis bewährt es sich sehr gut. Benutzer können parallel arbeiten, müssen niemals aufeinander warten. Wenn sie an denselben Dateien arbeiten, stellt sich heraus, dass die meiste Zeit keine Konflikte auftreten. Und die Zeit, die benötigt wird, um einen der wirklich seltenen Konflikte zu lösen, ist viel kleiner als die Zeit, welche auf die Freigabe einer gesperrten Datei gewartet wird.

Schlussendlich hängt alles von einem einzigen kritischen Faktor ab: Kommunikation unter den Benutzern. Wenn die Benutzer nur schlecht miteinander kommunizieren, werden vermehrt Konflikte auftreten. Kein System kann Benutzer zur Kommunikation zwingen, und kein System kann logische Konflikte erkennen.

Es gibt eine Situation, in der das Sperren-Ändern-Freigeben Modell Vorteile bietet. Das ist bei Dateien der Fall, die sich nicht automatisch zusammenführen lassen. Wenn Ihr Projektarchiv zum Beispiel Grafiken enthält und zwei Personen gleichzeitig dieselbe Grafik ändern, kann Subversion nicht entscheiden, wie die Änderungen zusammengeführt werden. Entweder Harry oder Sally wird seine Änderungen verlieren.

2.2.4. Was macht Subversion?

Subversion verwendet standardmäßig das Kopieren-Ändern-Zusammenführen Modell, und in den meisten Fällen wird das alles sein, was Sie jemals benötigen werden. Mit der Einführung von Version 1.2 unterstützt Subversion auch das Sperren von Dateien, so dass Ihnen diese Möglichkeit ebenfalls zur Verfügung steht.

2.3. Subversion bei der Arbeit

2.3.1. Arbeitskopien

Sie haben bereits über Arbeitskopien gelesen; nun werden wir zeigen, wie Subversion diese erstellt und benutzt.

Eine Subversion-Arbeitskopie ist eine ganz normale Ordnerstruktur auf einem lokalen Dateisystem, welches eine Anzahl von Dateien enthält. Sie können diese Dateien ändern wie Sie wollen, und falls dies Quellcode-Dateien sind, können Sie diese kompilieren wie sonst auch. Ihre Arbeitskopie ist Ihr eigenes privates Arbeitsfeld: Subversion wird nie Änderungen von anderen einfügen oder Ihre Änderungen anderen zur Verfügung stellen, ohne dass Sie Subversion explizit dazu auffordern.

Nachdem Sie einige Änderungen an den Dateien in Ihrer Arbeitskopie vorgenommen haben und Sie überprüft haben, dass diese immer noch korrekt funktionieren (z. B. ohne Fehler kompilierbar sind), stellt Ihnen Subversion Befehle zur Verfügung, um Ihre Änderungen zu *publizieren* (durch Speichern im zentralen Projektarchiv), d. h., anderen zur Verfügung zu stellen. Falls andere Benutzer ihre Änderungen veröffentlicht haben, stellt Subversion Befehle zur Verfügung, mit denen Sie diese Änderungen in Ihre Arbeitskopie einfügen können (durch Lesen aus dem zentralen Projektarchiv).

Eine Arbeitskopie enthält immer auch zusätzliche Dateien, erstellt und verwaltet von Subversion, um diese Befehle ausführen zu können. Genauer gesagt enthält jeder Ordner in Ihrer Arbeitskopie einen Unterordner mit Namen `.svn`, auch bekannt als der *Administrative Ordner* der Arbeitskopie. Die Dateien in jedem administrativen Ordner helfen Subversion dabei zu erkennen, welche Dateien Sie verändert haben und welche Dateien sich im Projektarchiv durch andere Benutzer geändert haben.

Ein typisches Subversion Projektarchiv enthält oft Dateien (oder Quellcode) von mehreren Projekten; normalerweise ist jedes Projekt ein Unterordner im Dateibaum. In einem solchen Umfeld enthält eine Arbeitskopie normalerweise nur einen solchen Unterordner des Projektarchivs.

Als Beispiel nehmen Sie an, dass Sie ein Projektarchiv haben, das zwei Softwareprojekte enthält.

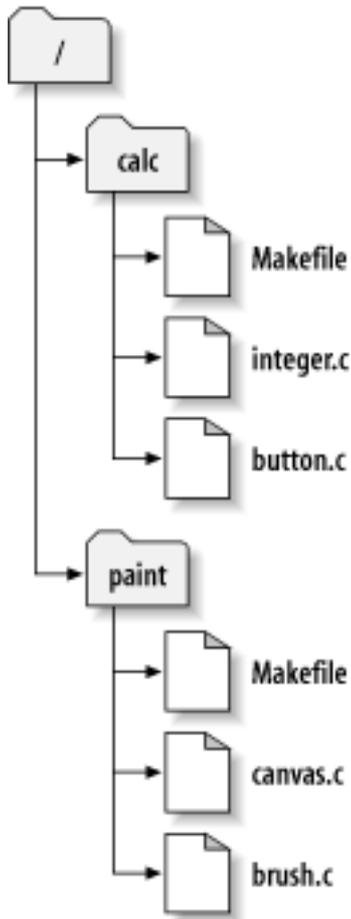


Abbildung 2.6. Das Dateisystem des Projektarchivs

In anderen Worten, das Hauptverzeichnis des Projektarchivs hat zwei Unterordner: `paint` und `calc`

Um eine Arbeitskopie zu erhalten, müssen Sie einen dieser Unterordner aus dem Projektarchiv *Auschecken*. Das klingt so, als würden dadurch Ressourcen gesperrt oder reserviert, aber dem ist nicht so. Es wird lediglich eine private Kopie des Projekts für Sie erzeugt.

Angenommen, Sie machen Änderungen an der Datei `button.c`. Da der `.svn` Ordner den Originalinhalt und das Änderungsdatum gespeichert hat, kann Subversion feststellen, dass Sie diese Datei geändert haben. Jedoch wird Subversion diese Änderungen erst veröffentlichen, wenn Sie Subversion dazu explizit auffordern. Der Vorgang des Veröffentlichens ist allgemein unter dem Begriff *Übertragen* (engl. `committing`) bekannt.

Um Ihre Änderungen anderen zur Verfügung zu stellen, verwenden Sie den Befehl **Übertragen**.

Danach sind Ihre Änderungen an der Datei `button.c` im Projektarchiv abgespeichert; falls jemand anders nun eine neue Arbeitskopie von `/calc` auscheckt, wird diese Ihre letzten Änderungen an der Datei `button.c` mit enthalten.

Nehmen Sie an, Sie haben eine Mitarbeiterin, Sally, die eine Arbeitskopie von `/calc` zur selben Zeit wie Sie ausgecheckt hat. Wenn Sie Ihre Änderungen übertragen, bleibt die Arbeitskopie von Sally unverändert. Subversion ändert an einer Arbeitskopie nur dann etwas, wenn Sie es ausdrücklich dazu auffordern.

Um ihre Arbeitskopie auf den neuesten Stand zu bringen, sagt Sally Subversion, dass es ihre Arbeitskopie *aktualisieren* soll mittels des **Aktualisieren** Befehls. Dies wird Ihre Änderungen in die Arbeitskopie von Sally einfügen, sowie natürlich alle weiteren Änderungen, welche andere vorgenommen haben seit Sally ihre Arbeitskopie das letzte mal aktualisiert hat.

Beachten Sie, dass Sally nicht angeben muss, welche Dateien genau Subversion aktualisieren soll; Subversion nutzt die Informationen im `.svn` Ordner und weitere Informationen aus dem Projektarchiv, um dies selbst festzustellen.

2.3.2. Projektarchiv URLs

Auf Subversion Projektarchive kann mittels vieler verschiedener Methoden zugegriffen werden - lokal oder durch verschiedene Netzwerk-Protokolle. Auf ein Objekt im Projektarchiv wird immer über eine URL zugegriffen. Das URL Schema zeigt die Zugriffsmethode:

Schema	Zugriffsmethode
<code>file://</code>	Direkter Zugriff auf einer lokalen Festplatte oder einem Netzwerkverzeichnis.
<code>http://</code>	Zugriff via WebDAV Protokoll auf einen Apache-Server, auf dem Subversion installiert ist.
<code>https://</code>	Wie <code>http://</code> , aber mit SSL-Verschlüsselung.
<code>svn://</code>	Nicht authentifizierter TCP/IP Zugriff mittels eigenem Protokoll auf einen <code>svnserve</code> Server.
<code>svn+ssh://</code>	Authentifizierter, verschlüsselter TCP/IP Zugriff mittels eigenem Protokoll auf einen <code>svnserve</code> Server.

Tabelle 2.1. Projektarchiv Zugriffs-URLs

In den weitaus meisten Fällen nutzen Subversion URLs die Standardsyntax und erlauben, dass Servernamen und Portnummern als Teil der URL angegeben werden. Beachten Sie jedoch, dass die `file://` Zugriffsmethode normalerweise nur für den Zugriff auf lokale Projektarchive verwendet wird, obwohl auch mittels eines UNC Pfades auf ein Projektarchiv im Netzwerk zugegriffen werden kann. In diesem Fall nimmt die URL die Form `file:///rechnername/pfad/zum/projektarchiv` an. Bei einem lokalen Projektarchiv muss der Rechnername entweder leer oder `localhost` sein. Deshalb enthalten lokale Pfade meistens drei Schrägstriche, `file:///pfad/zum/projektarchiv`.

Zudem müssen Benutzer des `file://` Schemas auf Windowsrechnern eine nicht offizielle „Standard“ Syntax für die URL verwenden. Jede der folgenden URLs funktioniert, wobei X das Laufwerk mit dem Projektarchiv ist:

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

Beachten Sie, dass die URL normale Schrägstriche verwendet, obwohl Windows-Pfade mit umgekehrten Schrägstrichen angegeben werden.

Sie können auf FSFS Projektarchive auf einer Netzwerkfreigabe zugreifen, für BDB Projektarchive ist das *nicht* möglich.



Warnung

Erstellen Sie kein Berkeley DB Projektarchiv auf einer Netzwerkfreigabe oder greifen auf eine solche zu. Eine BDB *darf nicht* auf einer Netzwerkfreigabe sein! Auch nicht

wenn die Netzwerkfreigabe als Laufwerk verbunden ist. Wenn Sie versuchen ein solches Projektarchiv zu verwenden wird dies zu unvorhersehbaren Effekten führen - Sie können mysteriöse Fehlermeldungen gleich zu Beginn erhalten, oder es können Monate vergehen ehe Sie feststellen dass das Projektarchiv korrupt und nicht mehr zu retten ist.

2.3.3. Revisionen

Ein **Übertragen** Befehl kann Änderungen an beliebig vielen Dateien und Ordnern als eine einzige atomare Operation übertragen. In Ihrer Arbeitskopie können Sie Dateien ändern, erstellen, löschen, umbenennen und verschieben und dann alle diese Änderungen in einem einzigen Schritt übertragen.

Im Projektarchiv wird jede Übertragung als eine einzige atomare Transaktion behandelt: entweder werden alle Änderungen abgespeichert oder gar keine. Subversion stellt dies sogar im Falle von Systemabstürzen, Netzwerkproblemen, Stromausfällen oder anderen Problemen sicher.

Jedes mal wenn ein Projektarchiv eine Übertragung akzeptiert, wird dadurch ein neuer Zustand der Dateistruktur erzeugt. Ein solcher Zustand wird *Revision* genannt. Jeder Revision wird eine natürliche Zahl zugeordnet, jeweils um eins größer als die vorherige Revision. Der Ursprungszustand eines Projektarchivs bei der Erstellung ist also Revision 0. Revision 0 enthält nichts außer einem leeren Hauptverzeichnis.

Ein möglicher Weg, um sich das Projektarchiv vorzustellen, ist als eine Reihe von Bäumen. Stellen Sie sich eine Reihe von Revisionsnummern vor, beginnend mit 0, sich von links nach rechts ausbreitend. Jede Revisionsnummer hat einen Dateibaum, welcher an dieser Reihe hängt, und jeder dieser Bäume ist eine „Momentaufnahme“ davon, wie das Projektarchiv nach der jeweiligen Übertragung ausgesehen hat.

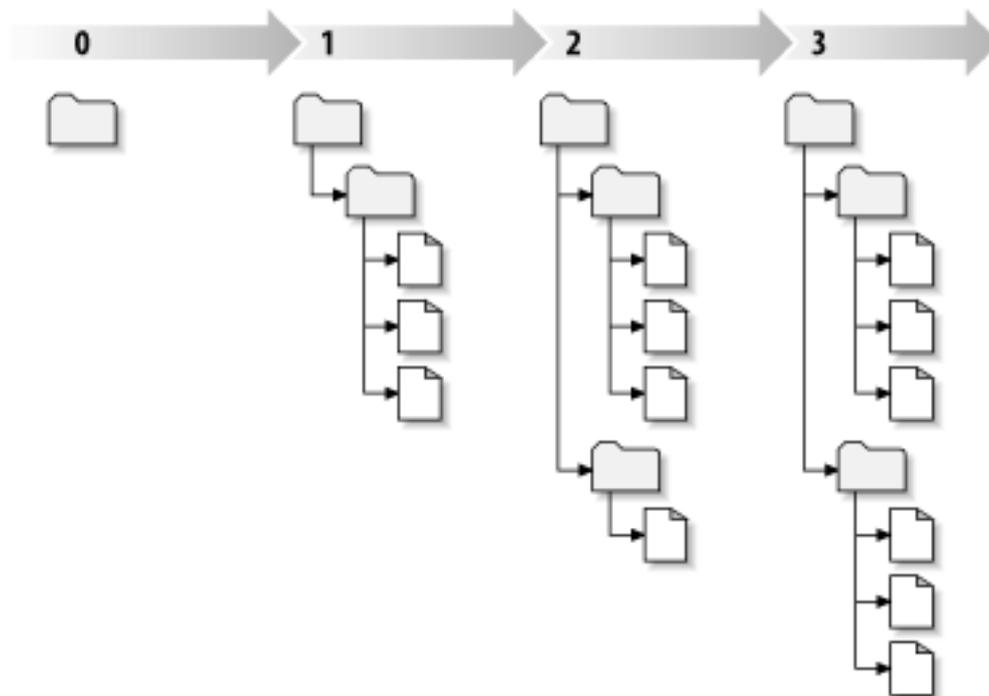


Abbildung 2.7. Das Projektarchiv

Globale Revisionsnummern

Anders als in vielen bekannten Versionskontrollsystemen stellt eine Revisionsnummer den Zustand eines *ganzen Dateibaumes* dar, nicht nur von einzelnen Dateien. Jede Revision ist ein ganzer Dateibaum, ein bestimmter Zustand des Projektarchivs nach einer Übertragung. Eine andere Möglichkeit, sich eine Revision vorzustellen, ist, dass eine Revision N den Zustand

des Projektarchivs nach der N-ten Übertragung darstellt. Wenn ein Subversion-Benutzer von „Revision 5 der Datei `foo.c`“ spricht, meint er in Wirklichkeit den Zustand der Datei `foo.c` wie sie in Revision 5 ausgesehen hat. Beachten Sie, dass zwei verschiedene Revisionen einer Datei *nicht* unterschiedlich sein müssen!

Es ist wichtig zu beachten, dass Arbeitskopien nicht immer mit einer einzigen Revision des Projektarchivs übereinstimmen; sie können Dateien von verschiedenen Revisionen enthalten. Nehmen Sie zum Beispiel an, Sie haben aus einem Projektarchiv eine Arbeitskopie ausgecheckt, bei welcher die neueste Revision die 4 ist:

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

In diesem Moment entspricht die Arbeitskopie exakt der Revision 4 im Projektarchiv. Nun nehmen Sie an, Sie ändern die Datei `button.c` und übertragen diese Änderungen. Nehmen Sie weiter an, dass keine weiteren Änderungen vorgenommen wurden und Ihre Übertragung eine neue Revision 5 im Projektarchiv erstellt. Ihre Arbeitskopie wird dann so aussehen:

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

Nehmen Sie nun an, dass jetzt Sally Änderungen an der Datei `integer.c` überträgt und damit eine Revision 6 erstellt. Wenn Sie nun Ihre Arbeitskopie aktualisieren, wird dies dann so aussehen:

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

Sallys Änderungen an der Datei `integer.c` werden in Ihre Arbeitskopie eingefügt, und Ihre Änderungen in der Datei `button.c` werden immer noch vorhanden sein. In diesem Beispiel ist der Inhalt der Datei `Makefile` identisch in den Revisionen 4, 5 und 6, aber Subversion markiert Ihre lokale Kopie von `Makefile` mit Revision 6 um zu zeigen, dass diese immer noch aktuell ist. Also nachdem Sie eine saubere Aktualisierung durchgeführt haben, wird Ihre Arbeitskopie exakt einer Revision im Projektarchiv entsprechen.

2.3.4. Wie Arbeitskopien das Projektarchiv verfolgen

Für jede Datei in einer Arbeitskopie zeichnet Subversion zwei wesentliche Informationen im `.svn` Administrationsordner auf:

- Auf welcher Revision die Datei basiert (dies wird auch *Arbeitsrevision* genannt)
- Der Zeitpunkt, an dem die Datei zum letzten Mal aus dem Projektarchiv aktualisiert wurde.

Mittels dieser Informationen ist Subversion in der Lage, durch Anfragen an das Projektarchiv herauszufinden, in welchem der folgenden vier Zustände eine Datei ist:

Unverändert und aktuell

Die Datei wurde weder lokal noch im Projektarchiv verändert. Eine **Übertragung** und auch eine **Aktualisierung** dieser Datei bewirkt nichts.

Lokal verändert und aktuell

Die Datei wurde lokal verändert, aber nicht im Projektarchiv. Eine **Übertragung** bewirkt ein Speichern dieser Änderungen im Projektarchiv. Eine **Aktualisierung** hingegen bewirkt nichts.

Unverändert, aber nicht mehr aktuell

Die Datei wurde lokal nicht verändert, jedoch gibt es Änderungen an der Datei im Projektarchiv. Eine **Übertragung** bewirkt nichts, jedoch wird ein **Aktualisieren** die Änderungen aus dem Projektarchiv in die lokale Kopie der Datei einfügen.

Lokal verändert und nicht aktuell

Die Datei wurde lokal und im Projektarchiv verändert. Ein **Übertragen** wird mit einer *nicht mehr aktuell* Fehlermeldung fehlschlagen. Sie müssen die Datei zuerst **Aktualisieren**, um die Änderungen aus dem Projektarchiv in die lokale Kopie einzufügen. Falls Subversion nicht in der Lage sein sollte, diese Änderungen selbst zusammenzuführen, was selten der Fall ist, muss der Benutzer den Konflikt manuell lösen.

2.4. Zusammenfassung

Wir haben eine Anzahl von grundlegenden Subversion-Konzepten in diesem Kapitel behandelt:

- Wir haben den Begriff des zentralen Projektarchivs, der Arbeitskopie und der Revisionen eingeführt.
- Wir haben einige einfache Beispiele gezeigt, wie zwei Mitarbeiter Subversion mittels des Kopieren-Ändern-Zusammenführen Modells benutzen können, um Änderungen zur Verfügung zu stellen und Änderungen von anderen zu übernehmen.
- Wir sprachen ein wenig darüber, wie Subversion Informationen in der Arbeitskopie verfolgt und verwaltet.

Kapitel 3. Das Projektarchiv

Egal welches Protokoll Sie verwenden, um auf das Projektarchiv zuzugreifen, Sie müssen jeweils mindestens ein Projektarchiv erstellen. Sie können dies entweder mit dem Subversion Kommandozeilen-Client oder mit TortoiseSVN tun.

Wenn Sie noch kein Subversion Projektarchiv erstellt haben, tun Sie dies jetzt.

3.1. Projektarchiv erstellen

Sie können ein Projektarchiv im FSFS Format oder im älteren Berkeley Database (BDB) Format erstellen. Das FSFS Format ist generell schneller, einfacher zu administrieren und es funktioniert auch auf Netzwerkfreigaben und Windows 98 ohne Probleme. Das BDB Format wurde als stabiler erachtet, weil es bereits länger in Gebrauch war, da aber FSFS mittlerweile auch schon mehrere Jahre intensiv genutzt wird, ist dieses Argument etwas schwach. Nähere Informationen zu den verschiedenen Formaten können Sie im Kapitel *Choosing a Data Store* [<http://svnbook.red-bean.com/en/1.5/svn.reposadmin.planning.html#svn.reposadmin.basics.backends>] im Subversion Buch nachlesen.

3.1.1. Ein Projektarchiv mit dem Kommandozeilen-Client erstellen

1. Erstellen Sie einen leeren Ordner mit dem Namen `SVN` (z.B. `D:\SVN\`), in welchem alle Projektarchive gespeichert werden.
2. Erstellen Sie einen weiteren Ordner `MyNewRepository` innerhalb von `D:\SVN\`.
3. In der Kommandozeile (auch DOS-Box genannt) wechseln sie zu `D:\SVN\` und geben sie folgendes ein:

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

oder

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

Nun haben Sie ein neues Projektarchiv in `D:\SVN\MyNewRepository` erstellt.

3.1.2. Erstellen eines Projektarchivs mit TortoiseSVN

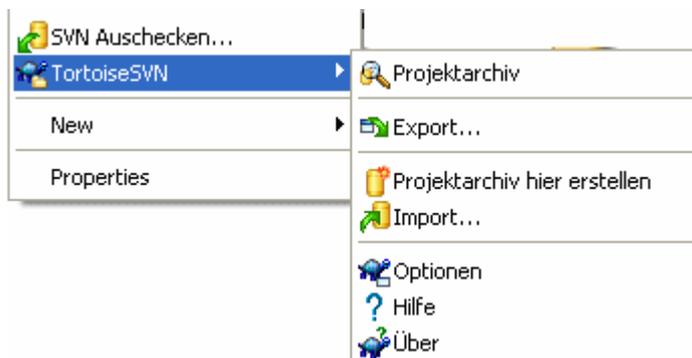


Abbildung 3.1. Das TortoiseSVN Menü für unversionierte Ordner

1. Öffne den Windows-Explorer
2. Erstellen Sie einen neuen Ordner und nennen Sie ihn zum Beispiel `SVNRepository`.

3. Rechtsklick auf einen neu erstellten Ordner und wählen Sie TortoiseSVN → Projektarchiv hier erstellen....

Ein Projektarchiv wird dann in dem neuen Ordner erstellt. *Editieren Sie diese Dateien nicht selbst.* Falls Sie eine Fehlermeldung erhalten stellen Sie sicher, dass der Ordner leer und nicht schreibgeschützt ist.



Tipp

TortoiseSVN unterstützt das Anlegen von BDB Projektarchiven nicht mehr. Sie können das jedoch weiterhin mit der SVN Kommandozeile tun. FSFS Projektarchive sind im Allgemeinen für den Anwender einfacher zu warten und machen es uns auch einfacher TortoiseSVN zu pflegen, da es Kompatibilitätsprobleme zwischen verschiedenen BDB Versionen gibt.

Zukünftige Versionen von TortoiseSVN werden den direkten `file://` Zugriff auf BDB Projektarchive aufgrund dieser Kompatibilitätsprobleme nicht mehr unterstützen. Der Zugriff auf BDB Projektarchive auf einem Server über die `svn://`, `http://` oder `https://` Protokolle wird jedoch weiterhin unterstützt. Aus diesem Grund empfehlen wir, dass Projektarchive, auf die mittels `file://` zugegriffen werden soll, nur noch im FSFS Format angelegt werden.

Grundsätzlich empfehlen wir allerdings auf die Verwendung des `file://` Zugriffs, außer für lokale Testzwecke, zu verzichten. Die Verwendung eines Servers ist, außer vielleicht für Einzelanwender, generell sicherer und zuverlässiger.

3.1.3. Lokaler Zugriff auf das Projektarchiv

Um das lokale Projektarchiv anzusprechen, benötigen Sie den Pfad zum Projektarchiv-Ordner. Beachten Sie, dass Subversion alle Projektarchivpfade in der Form `file:///C:/SVNRepository/` benötigt.

Um ein Projektarchiv auf einer Netzwerkfreigabe anzusprechen können Sie entweder die Freigabe als Laufwerk verbinden oder den UNC-Pfad benutzen. UNC-Pfade haben die Form `file://ServerName/Pfad/zum/Projektarchiv/`. Beachten Sie, dass hier nur zwei Schrägstriche vor dem Servernamen stehen.

Vor Subversion 1.2, mussten UNC Pfade in dem obskuren Format `file:///\\ServerName/path/to/repos` angegeben werden. Dieses Format wird weiterhin unterstützt, aber nicht empfohlen.



Warnung

Erstellen Sie kein Berkeley DB Projektarchiv auf einer Netzwerkfreigabe oder greifen auf eine solche zu. Eine BDB *darf nicht* auf einer Netzwerkfreigabe sein! Auch nicht wenn die Netzwerkfreigabe als Laufwerk verbunden ist. Wenn Sie versuchen ein solches Projektarchiv zu verwenden wird dies zu unvorhersehbaren Effekten führen - Sie können mysteriöse Fehlermeldungen gleich zu Beginn erhalten, oder es können Monate vergehen ehe Sie feststellen dass das Projektarchiv korrupt und nicht mehr zu retten ist.

3.1.4. Projektarchiv auf einer Netzwerkfreigabe

Obwohl es theoretisch möglich ist, ein FSFS Projektarchiv auf einer Netzwerkfreigabe anzulegen und mehrere Anwender per `file://` Protokoll darauf zugreifen zu lassen, raten wir *dringend* davon ab.

Erstens geben Sie jedem Anwender direkten Schreibzugriff auf das Projektarchiv, so dass jeder Benutzer das Projektarchiv versehentlich löschen oder anderweitig unbrauchbar machen kann.

Zweitens unterstützen nicht alle Netzwerk Dateiprotokolle die Sperr- und Schutzmechanismen, die Subversion erfordert, so dass das Projektarchiv beschädigt werden kann. Das muss nicht sofort passieren, aber eines Tages werden zwei Anwender gleichzeitig auf das Projektarchiv zugreifen. Schlimmstenfalls bleibt die Beschädigung eine Zeit lang unbemerkt.

Drittes müssen die Dateizugriffsrechte stimmen. Auf einer reinen Windowsfreigabe können Sie sich vielleicht durchmogeln, aber SAMBA ist besonders schwierig einzurichten.

`file://` Zugriff ist ausschließlich für lokalen Einzelbenutzerzugriff gedacht, speziell zum Testen und Debuggen. Wenn Sie das Projektarchiv gemeinsam nutzen wollen sollten Sie *wirklich* einen richtigen Server einrichten, und das ist nicht halb so kompliziert wie Sie vielleicht denken. Lesen Sie [Abschnitt 3.5, „Zugriff auf das Projektarchiv“](#) als Anleitung zur Auswahl und Einrichtung eines Servers.

3.1.5. Struktur des Projektarchivs

Bevor Sie Daten in das Projektarchiv importieren sollten Sie sich Gedanken darüber machen wie Sie Ihre Daten organisieren wollen. Wenn Sie eine der empfohlenen Strukturen verwenden, werden Sie es später sehr viel einfacher haben.

Es gibt ein paar empfohlene Standards um die Daten innerhalb eines Projektarchivs zu organisieren. Die meisten Leute erstellen einen `trunk` Ordner, der den „Stamm der Entwicklung“ enthält, einen `branches` Ordner, welcher Entwicklungszweige enthält und einen `tags` Ordner, der markierte Zustände enthält. Wenn ein Projektarchiv nur ein einziges Projekt enthält ist die folgende Struktur zu empfehlen:

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

Falls ein Projektarchiv jedoch mehrere Projekte enthält, wird innerhalb der Struktur nach Zweigen (`branches`) organisiert:

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

...oder nach Projekt:

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

Nach Projekt zu organisieren ist sinnvoll, wenn diese nicht eng miteinander verwandt sind und jedes für sich bearbeitet wird. Für verwandte Projekte, die sie alle in einem Rutsch auschecken wollen oder die in einer Distribution zusammengefasst sind, ist es oft besser nach Zweigen zu organisieren. Auf diese Weise müssen Sie nur den Stamm (`trunk`) auschecken und die Beziehungen zwischen den Unterprojekten sind einfacher zu erkennen.

Wenn Sie eine `/trunk /tags /branches` Struktur wählen, müssen Sie lediglich den gesamten Stamm kopieren, um Zweige oder Marken zu erstellen. In vielen Punkten ist das der flexibelste Ansatz.

Für völlig unabhängige Projekte sollten Sie besser pro Projekt ein einzelnes Projektarchiv anlegen. Der Grund dafür ist, dass wenn Sie Änderungen übertragen, die Revisionsnummer des Projektarchivs, nicht des Projektes hoch gezählt wird. Bei mehreren unabhängigen Projekten, die sich ein Archiv teilen, kann das zu Konfusion führen, wenn z.B. an einem der Projekte überhaupt nicht gearbeitet wird, an anderen jedoch viel. TortoiseSVN und Subversion zum Beispiel, erscheinen auf demselben Server (tigris.org), sind aber völlig selbstständige Projekt(archiv)e, mit voneinander unabhängiger Entwicklung und ohne Verwirrung über Versionsnummern.

Natürlich können Sie diese Empfehlungen auch ignorieren. Es steht Ihnen frei, ein Layout zu wählen welches am besten für Sie oder Ihr Team geeignet ist. Beachten Sie auch, dass was immer für eine Wahl Sie treffen diese nicht fix ist. Sie können das Layout später immer noch reorganisieren. Weil Zweige (branches) und Marken (tags) gewöhnliche Ordner in Subversion sind, können Sie diese ganz einfach mit TortoiseSVN umbenennen oder verschieben wie Sie wollen.

Von einem Layout zu einem anderen zu wechseln ist lediglich eine Abfolge serverseitiger Umbenennungen; Wenn Ihnen die Art und Weise wie Dinge im Projektarchiv organisiert sind, nicht gefallen, jonglieren Sie einfach mit den Verzeichnissen herum.

Falls Sie noch keine Ordnerstruktur in Ihrem Projektarchiv angelegt haben, sollten Sie das jetzt erledigen. Es gibt zwei Möglichkeiten dafür. Falls Sie nur die einfache `/trunk /tags /branches` Struktur wünschen, können Sie diese in drei Schritten mit Hilfe des Projektarchivbetrachters anlegen. Wenn Sie eine tiefer verschachtelte Hierarchie wünschen, sollten Sie die zunächst auf der lokalen Platte anlegen und dann, wie hier, in einem Schritt importieren :

1. Erstellen Sie einen leeren Ordner auf Ihrer Festplatte
2. Erstellen Sie in diesem Ordner die gewünschte Ordnerstruktur - kopieren Sie noch keine Dateien hinein!
3. Importieren Sie diese Struktur durch Rechtsklick auf den obersten Ordner und Klick auf TortoiseSVN → Importieren... in das Projektarchiv. Dies importiert Ihren Ordner in das Projektarchiv und erstellt dort die gewünschte Ordnerstruktur.

Beachten Sie dass der Name des Ordners welchen Sie importieren *nicht* im Projektarchiv erscheint, nur dessen Inhalt. Erstellen Sie zum Beispiel folgende Ordnerstruktur:

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

Importieren Sie C:\Temp\New ins Projektarchiv, welches dann so aussieht:

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

3.2. Projektarchiv sichern

Welches Projektarchivformat Sie auch immer verwenden, es ist sehr wichtig regelmäßige Datensicherungen vorzunehmen und auch dass Sie die gesicherten Daten überprüfen. Wenn der Server ausfällt und das Projektarchiv korrupt ist, haben Sie zwar noch die Arbeitskopie mit der letzten Version - aber ohne ein Backup verlieren Sie die gesamte Historie des Projekts!

Der einfachste (jedoch nicht empfohlene) Weg ein Backup zu erstellen ist die Projektarchiv-Dateien einfach auf ein Backup-Medium zu kopieren. Wenn Sie dies tun, müssen Sie absolut sicher sein, dass während des Kopiervorgangs niemand auf das Projektarchiv zugreift. Das bedeutet wirklich *jeden*

Zugriff. Auf ein Projektarchiv wird sogar dann geschrieben, wenn die eigentliche Operation nur lesend ist wie zum Beispiel eine Statusabfrage. Wird auf das Projektarchiv während des Kopiervorgangs zugegriffen (Web-Browser offen, WebSVN, Log Anzeigen, ...) dann ist die Kopie (und damit das Backup) wertlos.

Die empfohlene Methode ist ein

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

auszuführen um eine Kopie des Projektarchivs zu erstellen. Dann können Sie diese Kopie sichern (z.B. auf Band). Die `--clean-logs` Option ist nicht notwendig, entfernt aber redundante (und damit nicht unbedingt notwendige) Log-Dateien wenn Sie die Sicherung erstellen und hilft somit Platz zu sparen.

Das `svnadmin` Programm wird automatisch installiert wenn Sie den Subversion Kommandozeilen-Client installieren. Wenn Sie die Kommandozeilenprogramme auf einem Windows-PC installieren ist es am besten, die Windows-Installerversion zu verwenden. Diese ist besser komprimiert als die `zip`-Version und nimmt auch alle Pfad-Einstellungen für Sie vor. Sie können die jeweils aktuellste Version hier holen: <http://subversion.apache.org/getting.html>.

3.3. Serverseitige Aktionskripte

Ein Aktionskript (Hook-Script) ist ein Programm, welches durch bestimmte Ereignisse im Projektarchiv gestartet wird, z.B. wenn eine neue Revision erzeugt oder eine Revisions-Eigenschaft verändert wird. Jedem Skript werden Information über das Ereignis selbst, die Ziele des Ereignisses und der Benutzername der Person welche das Ereignis ausgelöst hat. Abhängig vom Rückgabewert des Aktionskripts wird die Operation fortgesetzt, angehalten oder unterbrochen. Bitte lesen Sie dazu das Kapitel *Hook Scripts* [<http://svnbook.red-bean.com/en/1.5/svn.reposadmin.create.html#svn.reposadmin.create.hooks>] im Subversion Buch.

Diese Aktionskripte werden auf dem Server ausgeführt, der das Projektarchiv beherbergt. TortoiseSVN bietet Ihnen obendrein die Möglichkeit, lokale Aktionskripte bei bestimmten Ereignissen auszuführen. Lesen Sie in [Abschnitt 4.30.8, „Clientseitige Aktionskripte“](#) nach, wie das geht.

Beispiele für Aktionskripte findet man im `hooks` Ordner des Projektarchivs. Diese Beispiele sind für Unix/Linux-Server erstellt worden und müssen ein wenig angepasst werden wenn Ihr Server unter Windows läuft. Das Aktionskript kann eine einfache Batch-Datei oder ein ausführbares Programm sein. Das unten stehende Beispiel zeigt eine Batch-Datei welche als `pre-revprop-change` Aktion eingesetzt werden kann.

```
rem Nur Änderungen an Logmeldungen zulassen.
if "%4" == "svn:log" exit 0
echo Eigenschaft '%4' kann nicht geändert werden >&2
exit 1
```

Beachten Sie dass alle Ausgaben via `stdout` verworfen werden. Wenn Sie möchten, dass eine Fehlermeldung auch in TortoiseSVN angezeigt wird, müssen Sie diese Fehlermeldung via `stderr` ausgeben. In einer Batchdatei erreichen Sie dies über `>&2`.

3.4. Auschecken aus Webseiten

Wenn Sie Ihr Subversion Projektarchiv anderen Entwicklern zugänglich machen wollen, möchten Sie vielleicht von Ihrer Webseite aus direkt darauf verweisen. Eine Möglichkeit besteht darin, einen *Auscheck Verweis* für andere TortoiseSVN Nutzer dort zu platzieren.

Wenn Sie TortoiseSVN installieren, registriert es ein neues `tsvn:` Protokoll auf Ihrem Rechner. Sobald ein Anwender auf solch einen Verweis klickt, öffnet sich der Auschecken-Dialog mit der bereits ausgefüllten URL des Projektarchivs.

Um einen solchen Verweis auf Ihrer Webseite zu platzieren, müssen Sie entsprechenden Code hinzufügen. Ein Beispiel:

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

Selbstverständlich würde es noch besser aussehen, wenn der Verweis ein passendes Bild enthielte. Sie können folgendes Bild verwenden [TortoiseSVN Logo](https://tortoisesvn.tigris.org/svn/tortoisesvn/trunk/doc/logo/TortoiseCheckout.png) [tsvn:https://tortoisesvn.tigris.org/svn/tortoisesvn/trunk/doc/logo/TortoiseCheckout.png] oder Ihr eigenes Bild angeben.

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

Sie können den Verweis auch auf eine spezifische Revision, z.B.

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

zeigen lassen.

3.5. Zugriff auf das Projektarchiv

Um TortoiseSVN (oder einen anderen Subversion Client) nutzen zu können, benötigen Sie einen Ort, an dem Ihre Projektarchive liegen. Sie können diese entweder lokal anlegen und mit dem `file://` Protokoll darauf zugreifen oder Sie können die Projektarchive auf einem Server anlegen und mit den `http://` oder `svn://` Protokollen darauf zugreifen. Die beiden Serverprotokolle können zusätzlich verschlüsselt werden. Dazu verwenden Sie entweder `https://`, `svn+ssh://` oder `svn://` mit SASL.

Wenn Sie einen öffentlichen Dienstleister, wie [Google Code](http://code.google.com/hosting/) [http://code.google.com/hosting/] benutzen oder Ihr Server bereits von jemand anderem eingerichtet wurde, gibt es nichts weiter zu tun. Blättern Sie weiter zu [Kapitel 4, Anleitung zum täglichen Gebrauch](#).

Wenn Ihnen kein Server zur Verfügung steht und Sie alleine arbeiten oder wenn Sie Subversion und TortoiseSVN isoliert testen wollen, sind lokale Projektarchive für Sie die beste Wahl. Legen Sie einfach, wie in [Kapitel 3, Das Projektarchiv](#) ein lokales Projektarchiv auf ihrem Rechner an. Sie können den Rest dieses Kapitels überspringen und direkt zu [Kapitel 4, Anleitung zum täglichen Gebrauch](#) gehen.

Falls Sie in Erwägung gezogen haben, ein Projektarchiv für mehrere Benutzer auf einer Netzwerkfreigabe anzulegen, sollten Sie dies noch einmal überdenken. Lesen Sie in [Abschnitt 3.1.4, „Projektarchiv auf einer Netzwerkfreigabe“](#) nach, warum wir glauben, dass das eine schlechte Idee ist. Einen Server einzurichten ist nicht so schwierig, wie es klingt und wird ihnen höhere Zuverlässigkeit und eventuell auch höhere Geschwindigkeit bieten.

Die nächsten Abschnitte enthalten eine schrittweise Anleitung, wie Sie einen Server unter Windows aufsetzen. Sie können natürlich auch einen Server unter Linux einrichten, aber das sprengt den Rahmen dieser Anleitung. Detaillierte Informationen über Subversion Serveroptionen und wie Sie die beste Architektur für Ihre Situation wählen, finden Sie in [Server Configuration](http://svnbook.red-bean.com/en/1.5/svn.serverconfig.html) [http://svnbook.red-bean.com/en/1.5/svn.serverconfig.html].

3.6. Svnserve als Server

3.6.1. Einleitung

Subversion enthält Svnserve, einen eigenständigen kleinen Server, der ein eigenes Protokoll auf einer Standard TCP/IP Verbindung nutzt. Er ist ideal für kleinere Installationen oder wo ein kompletter Apache nicht eingesetzt werden kann.

In den meisten Fällen ist svnservice einfacher aufzusetzen und arbeitet obendrein schneller als ein Apache basierter Server. Allerdings stehen einige fortgeschrittene Funktionen nicht zur Verfügung. Dadurch, dass mittlerweile SASL enthalten ist, kann der Server auch einfach abgesichert werden.

3.6.2. Svnservice installieren

1. Laden Sie die neueste Subversion Version von <http://subversion.apache.org/getting.html> herunter. Alternativ können Sie sich ein Installationsprogramm von CollabNet unter <http://www.collab.net/downloads/subversion> herunterladen. Dieses Installationsprogramm richtet svnservice als Windows Dienst ein und enthält obendrein einige der Werkzeuge, die Sie zum Einrichten von SASL zur Absicherung des Servers benötigen.
2. Wenn Sie bereits Subversion installiert haben und svnservice läuft, müssen Sie den Dienst stoppen, bevor Sie weitermachen
3. Starten Sie das Subversion Installationsprogramm. Wenn Sie die Installation auf Ihrem Server durchführen (empfohlen), können Sie Schritt 4 überspringen.
4. Öffnen Sie das Installationsverzeichnis von Subversion im Windows Explorer. (normalerweise C:\Programme\Subversion). Aus dem bin Verzeichnis kopieren Sie die Dateien svnservice.exe, intl3_svn.dll, libapr.dll, libapriconv.dll, libapriutil.dll, libdb*.dll, libeay32.dll und ssleay32.dll in das Verzeichnis in dem Ihr Server installiert ist z.B. c:\svnservice.

3.6.3. Svnservice starten

Nun, da svnservice installiert ist, müssen Sie die Anwendung auch aktivieren. Die einfachste Variante ist, das folgende Skript in der Eingabeaufforderung zu starten:

```
svnservice.exe --daemon
```

Sie können auch eine Verknüpfung dafür anzulegen. Svnservice wird nun auf Anfragen warten, die über Port 3690 eingehen. Die --daemon Option zwingt svnservice, so lange als Dienst zu laufen, bis der Prozess manuell beendet wird.

Wenn Sie noch kein Projektarchiv angelegt haben, befolgen Sie die folgenden Anweisungen dazu [Abschnitt 3.7.4, „Konfiguration“](#).

Um schnell herauszufinden, ob svnservice arbeitet, rufen Sie mittels TortoiseSVN → Projektarchiv den Projektarchivbetrachter aus dem Kontextmenü von TortoiseSVN auf.

Nehmen wir an, Ihr Projektarchiv befindet sich in c:\repos\TestRepo und Ihr Server heißt localhost. Dann müssen Sie in das URL Feld des Projektarchivbetrachters:

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

eingeben.

Sie können beim Eingeben von URLs Zeit sparen und die Sicherheit erhöhen, indem Sie svnservice mit der --root Option einen Basispfad für das Projektarchiv übergeben und den Zugriff auf ein bestimmtes Verzeichnis auf dem Server beschränken:

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

Mit dem vorigen Test als Vorgabe würde svnservice folgendermaßen aufgerufen:

```
svnservice.exe --daemon -r c:\repos
```

In TortoiseSVN würde sich die URL zum Projektarchiv dementsprechend verkürzen:

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

Beachten Sie, dass die `-r` Option auch verwendet werden muss, wenn sich das Projektarchiv auf einer anderen Partition oder einem anderen Laufwerk auf dem Server befindet, als `Svnserve`.

`Svnserve` kann eine beliebige Anzahl von Projektarchiven bereitstellen. Sie müssen diese nur unterhalb des soeben von Ihnen festgelegten Basisordners erzeugen und über eine zum Basisordner relative URL darauf zugreifen.



Warnung

Erstellen Sie kein Berkeley DB Projektarchiv auf einer Netzwerkfreigabe oder greifen auf eine solche zu. Eine BDB *darf nicht* auf einer Netzwerkfreigabe sein! Auch nicht wenn die Netzwerkfreigabe als Laufwerk verbunden ist. Wenn Sie versuchen ein solches Projektarchiv zu verwenden wird dies zu unvorhersehbaren Effekten führen - Sie können mysteriöse Fehlermeldungen gleich zu Beginn erhalten, oder es können Monate vergehen ehe Sie feststellen dass das Projektarchiv korrupt und nicht mehr zu retten ist.

3.6.3.1. Svnserve als Dienst starten

Es ist normalerweise nicht die beste Lösung, `svnserve` als Anwender laufen zu lassen. Das bedeutet nämlich, dass ständig ein Anwender an Ihrem Server angemeldet ist oder dass jemand `svnserve` versehentlich beenden beziehungsweise nach einem Neustart vergessen könnte den Prozess ebenfalls zu starten. Eine bessere Möglichkeit besteht darin, `svnserve` als Dienst zu starten. Beginnend mit Subversion 1.4 kann `svnserve` als Windows-Dienst gestartet werden, vorherige Versionen benötigen ein Zusatzprogramm.

Um `svnserve` als Dienst einzurichten, der gemeinsam mit Windows startet, führen Sie folgenden Befehl aus.

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

Die Eingabe des Befehls muss in einer einzigen Zeile erfolgen. Falls einer der Pfade Leerzeichen enthält, müssen Sie diesen folgendermaßen in Anführungszeichen einschließen

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

Sie können auch nach dem Start des Dienstes eine Beschreibung angeben. Diese wird im Windows Dienstemanager angezeigt.

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

Beachten Sie bitte das eher unübliche Kommandozeilenformat von `sc`. In den Schlüssel= Wert Paaren darf kein Leerzeichen zwischen dem Schlüssel und dem = stehen, vor dem Wert hingegen muss ein Leerzeichen stehen.



Tipp

Microsoft empfiehlt mittlerweile, dass Dienste entweder unter dem lokale Dienste- oder dem Netzwerkdienstkonto ausgeführt werden. Lesen Sie mehr dazu unter [The Services and](#)

Service Accounts Security Planning Guide [<http://www.microsoft.com/technet/security/topics/serversecurity/serviceaccount/default.mspx>]. Um den Dienst unter dem lokalen Dienstkonto einzurichten, fügen Sie den folgenden Text an das obenstehende Beispiel an.

```
obj="NT AUTHORITY\LocalService"
```

Beachten Sie, dass Sie dem lokalen Dienstkonto die entsprechenden Zugriffsberechtigungen sowohl auf Subversion als auch auf Ihre Projektarchive erteilen müssen. Das gilt auch für Anwendungen, die von Aktionskripten aufgerufen werden. Die interne Gruppe dafür heißt "LOCAL SERVICE".

Sobald Sie den Dienst installiert haben, müssen Sie ihn im Dienstemanager von Hand starten (nur dieses eine Mal; Der Dienst wird automatisch aktiviert sobald der Rechner neu startet).

Für weitere Informationen schauen Sie bitte in *Windows Service Support for Svnserve* [<http://svn.collab.net/repos/svn/trunk/notes/windows-service.txt>] nach.

Falls Sie eine ältere Version von svnserve mit Hilfe des SvnService Programmes gestartet haben und nun svnserve direkt verwenden wollen, sollten Sie svnservice als Dienst deinstallieren. Vorher müssen sie den Dienst allerdings anhalten! Verwenden Sie einfach den Befehl

```
svnservice -remove
```

um den Diensteintrag zu entfernen.

3.6.4. Einfache Anmeldung mit Svnserve

Die Standardkonfiguration von svnserve lässt nur anonymen Lesezugriff zu, so dass Sie zwar das Projektarchiv über eine `svn://` URL betrachten, auschecken und aktualisieren, jedoch keine weiteren Operationen wie übertragen vornehmen können.

Um Schreibzugriff auf das Projektarchiv zu ermöglichen, müssen Sie die `conf/svnserve.conf` Datei im Verzeichnis des Projektarchivs bearbeiten. Diese Datei enthält die Dokumentation sowie die Parameter zur Einrichtung des svnserve Dienstes.

Sie können anonymen Schreibzugriff ermöglichen, indem Sie einfach:

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

eintragen. Allerdings werden Sie nicht erfahren, wer Änderungen an Daten im Projektarchiv vorgenommen hat, weil die `svn:author` Eigenschaft leer bleibt. Obendrein können Sie nicht kontrollieren, wer Änderungen an Daten vornimmt, da Sie es ja explizit *jedem, anonym* gestatten!

Eine Möglichkeit, den Zugriff zu steuern, ist eine Passwortdatenbank anzulegen:

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

Wobei `userfile` im gleichen Verzeichnis wie `svnserve.conf` liegt. Diese Datei kann auch an einer anderen Stelle im Dateisystem liegen (nützlich, wenn Sie mehrere Projektarchive haben, die die gleichen Zugriffsrechte benötigen). In diesem Fall muss mit einer absoluten Pfadangabe oder mit einer Pfadangabe relativ zum `conf` Verzeichnis auf die Datei verwiesen werden. Wenn Sie einen Pfad angeben muss er `/` wie `/in/unix` geschrieben werden. Ein `\` oder ein Laufwerksbuchstabe funktionieren nicht. Die Datei `userfile` hat folgende Struktur:

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

Auf diese Weise wird allen nicht angemeldeten (anon-access) Benutzern der Zugriff verweigert, wogegen alle in `userfile` aufgeführten Anwender Schreibzugriff erhalten.



Tipp

Falls Sie mehrere Projektarchive mit derselben Passwortdatenbank verwalten, wird die Einrichtung eines Anmeldebereichs das Leben für die Anwender vereinfachen, weil TortoiseSVN Anmeldeinformationen speichern kann, so dass sie nur einmal eingegeben werden müssen. Weitere Informationen finden sich in folgenden Abschnitten des Subversion Buchs: [Create a 'users' file and realm](http://svnbook.red-bean.com/en/1.5/svn.serverconfig.svnservice.html#svn.serverconfig.svnservice.auth.users) [http://svnbook.red-bean.com/en/1.5/svn.serverconfig.svnservice.html#svn.serverconfig.svnservice.auth.users] und [Client Credentials Caching](http://svnbook.red-bean.com/en/1.5/svn.serverconfig.netmodel.html#svn.serverconfig.netmodel.credcache) [http://svnbook.red-bean.com/en/1.5/svn.serverconfig.netmodel.html#svn.serverconfig.netmodel.credcache].

3.6.5. Bessere Sicherheit mit SASL

3.6.5.1. Was ist SASL?

Beim Cyrus „Simple Authentication and Security Layer“ handelt es sich um eine Open Source Bibliothek, die von der Carnegie Mellon Universität geschrieben wurde. Sie fügt allgemeine Anmelde- und Verschlüsselungsfunktionen zu jedem Netzwerkprotokoll hinzu. Ab Subversion 1.5 unterstützen sowohl der svnservice Server und der TortoiseSVN Client diese Bibliothek.

Für eine vollständige Beschreibung der zur Verfügung stehenden Optionen sollten Sie sich im Subversion Buch das Kapitel [Using svnservice with SASL](http://svnbook.red-bean.com/en/1.5/svn.serverconfig.svnservice.html#svn.serverconfig.svnservice.sasl) [http://svnbook.red-bean.com/en/1.5/svn.serverconfig.svnservice.html#svn.serverconfig.svnservice.sasl] durchlesen. Wenn Sie nur an einem einfachen Verfahren interessiert sind, mit dessen Hilfe Sie sichere Anmeldung und Verschlüsselung auf einem Windows Server einrichten können, so dass auf Ihr Projektarchiv sicher über das große böse Internet zugegriffen werden kann, lesen Sie hier weiter.

3.6.5.2. SASL Anmeldung

Um die SASL Mechanismen auf Ihrem Server zu aktivieren, müssen Sie drei Dinge tun. Als Erstes erstellen Sie eine `[sasl]` Sektion in der `svnservice.conf` Datei Ihres Projektarchivs, die folgenden Eintrag enthält:

```
use-sasl = true
```

Zweitens legen Sie eine Datei namens `svn.conf` an geeigneter Stelle an - typischerweise im Installationsverzeichnis von Subversion.

Drittens erzeugen Sie zwei neue Registrierungseinträge, damit SASL weiß, wo seine Dateien stehen. Erstellen Sie einen Schlüssel namens `[HKEY_LOCAL_MACHINE\SOFTWARE\Carnegie Mellon\Project Cyrus\SASL Library]` und legen Sie dort zwei neue Zeichenketten an: `SearchPath` zeigt auf das Verzeichnis, das die `sasl*.dll` enthält (Normalerweise im Subversion Installationsverzeichnis), und `ConfFile` zeigt auf das Verzeichnis, das die `svn.conf` Datei enthält. Falls Sie den CollabNet Installer verwendet haben, wurden diese Einträge bereits erzeugt.

Editieren Sie die `svn.conf` Datei, so dass sie folgende Einträge enthält:

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

Die letzte Zeile enthält den Ort der Anmeldedatenbank, eine Datei namens `sasldb`. Diese Datei kann überall liegen, ein guter Platz ist jedoch der Elternpfad des Projektarchivs. Stellen Sie sicher, dass der svnservice Dienst Lesezugriff auf diese Datei hat.

Falls svnserve bereits läuft, müssen Sie den Dienst neu starten, damit er die geänderte Konfiguration einliest.

Nun, da alles eingerichtet ist, müssen Sie nur noch die Benutzer und Passwörter einrichten. Dazu benötigen Sie das saslpasswd2 Programm. Falls Sie den CollabNet Installer benutzt haben, findet sich die Anwendung im Installationsverzeichnis. Der Aufruf lautet wie folgt:

```
saslpasswd2 -c -f C:\TortoiseSVN\sasldb -u Anmeldebereich Name
```

Der -f Parameter gibt den Ort der Datenbank an, Anmeldebereich muss der gleiche Wert wie in der svnserve.conf Datei Ihres Projektarchivs sein und Name ist, wie erwartet, der Name des Anwenders. Beachten Sie bitte, dass der Anmeldebereich keine Leerzeichen enthalten darf.

Sie können die in der Datenbank gespeicherten Benutzer mit Hilfe des sasldblistusers2 Programms auflisten.

3.6.5.3. SASL Verschlüsselung

Um verschiedene Stufen der Verschlüsselung ein- und auszuschalten, können Sie zwei Werte in der svnserve.conf Datei Ihres Projektarchivs setzen:

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

Die min-encryption und max-encryption Variablen legen die vom Server verlangte Verschlüsselung fest. Um die Verschlüsselung komplett abzuschalten, setzen Sie beide Werte auf Null. Für einfache Prüfsummen ohne Verschlüsselung (Verhindern von Manipulationen und Sicherstellen der Datenintegrität), setzen Sie beide Werte auf Eins. Wenn Sie Verschlüsselung zulassen aber nicht erzwingen wollen, setzen Sie den Minimalwert auf Null und den Maximalwert auf eine Bitlänge. Um Verschlüsselung zu erzwingen, setzen Sie beide Werte auf Zahlen größer als Eins. In unserem obenstehenden Beispiel verlangen wir von den Clients mindestens 128-Bit Verschlüsselung aber nicht mehr als 256-Bit Verschlüsselung.

3.6.6. Anmeldung mit svn+ssh

Eine andere Möglichkeit Benutzer an einem svnserve basierten Server anzumelden, besteht darin, eine Secure Shell (SSH) zu installieren, durch die die Anfragen weitergeleitet (getunnelt) werden. Diese ist nicht so einfach einzurichten wie SASL, kann aber in manchen Fällen auch eine gute Lösung sein.

Mit diesem Ansatz läuft svnserve nicht als Dienst, sondern die Secure Shell startet svnserve mit den Berechtigungen des über SSH angemeldeten Benutzers. Dafür benötigen Sie allerdings einen SSH Dienst auf Ihrem Server.

Eine einfache Anleitung zum Einrichten Ihres Servers findet sich in [Anhang G, Svnserve mit SSH sichern](#). Sie können weitere Beiträge zum Thema SSH in der FAQ finden, indem Sie nach „SSH“ suchen.

Weitere Informationen über svnserve finden sich in [Versionskontrolle mit Subversion](#) [<http://svnbook.red-bean.com/>].

3.6.7. Pfadbasierte Autorisierung mit svnserve

Beginnend mit Subversion 1.3, unterstützt svnserve die gleiche pfadbasierte Autorisierung über mod_authz_svn wie Apache. Sie müssen die Datei conf/svnserve.conf im Projektarchivverzeichnis bearbeiten und eine Zeile, die auf die Berechtigungsdatei verweist, einfügen:

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

Bei `authz` handelt es sich um eine Datei, die Sie anlegen, um die Zugriffsberechtigungen zu definieren. Sie können dieselbe Datei für mehrere Projektarchive verwenden. Informationen über das Dateiformat finden sich in [Abschnitt 3.7.6, „Pfadbasierte Autorisierung“](#).

3.7. Apache als Server

3.7.1. Einleitung

Der vielseitigste aller Server für Subversion ist der auf Apache basierende Server. Obwohl er ein wenig schwieriger einzurichten ist, bietet er den anderen Varianten gegenüber einige Vorteile:

WebDAV

Der Apache-basierte Subversion Server nutzt das WebDAV Protokoll, das auch von vielen anderen Programmen unterstützt wird. Sie können ein solches Verzeichnis zum Beispiel als „Web-Ordner“ im Windows Explorer einbinden und wie jeden anderen Ordner im Dateisystem benutzen.

Das Projektarchiv betrachten

Sie können Ihren Web-Browser per `http://` auf die URL des Projektarchivs zeigen lassen und dessen Inhalt betrachten, ohne einen Subversion Client zu installieren. Dies ermöglicht einem großen Anwenderkreis auf Ihre Daten zuzugreifen.

Anmeldung

Sie können jedes Anmeldeverfahren nutzen, das von Apache unterstützt wird, zum Beispiel SSPI und LDAP.

Sicherheit

Da Apache sehr stabil und sicher ist, bekommen Sie damit automatisch die gleiche Sicherheit (zum Beispiel SSL Verschlüsselung) für Ihr Projektarchiv.

3.7.2. Apache installieren

Für die Installation von Apache unter Windows benötigen Sie zumindest Windows2000 / WinXP+SP1 oder Windows2003, Vista oder Server 2008.



Warnung

Bitte beachten Sie, dass Windows XP ohne Servicepack 1 zu fehlerhaften Netzwerkdaten führt und dadurch Ihr Projektarchiv beschädigt werden kann!

1. Laden Sie die neueste Version des Apache Web-Servers von <http://httpd.apache.org/download.cgi>. Achten Sie darauf, dass Sie Version 2.2.x oder neuer herunterladen - Die 1.3.xx arbeiten nicht mit Subversion zusammen!

Den MSI Installer für Apache finden Sie, indem Sie auf `other files` klicken und dann zu `binaries/win32` navigieren. Wenn Sie SSL benötigen sollten Sie `apache-2.2.x-win32-x86-openssl-0.9.x.msi` herunterladen.

2. Sobald Sie den Apache 2 Installer haben, können Sie ihn mit einem Doppelklick starten. Sie werden Schritt für Schritt durch den Installationsprozess geführt. Stellen Sie sicher, dass Sie die Server URL korrekt eingeben. Wenn Sie keinen DNS Server haben, nehmen Sie stattdessen einfach die IP Adresse. Wir raten dazu, Apache für *Alle Benutzer, auf Port 80, als Dienst* zu installieren. Achtung: Wenn Sie bereits IIS oder ein anderes Programm installiert haben, das auf Port 80 lauscht, kann die Installation fehlschlagen. Wenn das passiert, gehen Sie in `C:\Programme\Apache Group\Apache2\conf` und öffnen Sie dort die Datei `httpd.conf`. Suchen Sie den Eintrag `Listen 80` und ändern Sie diesen in eine freie Portnummer, z.B. `Listen 81`. Starten Sie die Installation erneut - Diesmal sollte sie ohne Probleme durchlaufen.

3. Testen Sie nun, ob Apache korrekt installiert ist, indem Sie in Ihren Web-Browser die URL `http://localhost/` eingeben. Eine Standard Webseite sollte angezeigt werden.



Achtung

Wenn Sie sich entscheiden, Apache als Dienst zu installieren, beachten Sie bitte dass er dann mit den lokalen SYSTEM Berechtigungen läuft. Es ist wesentlich sicherer, wenn Sie für Apache ein eigenes Benutzerkonto anlegen in dessen Kontext der Dienst dann läuft.

Stellen Sie sicher, dass das Konto unter dem der Apache Server läuft, einen eigenen Eintrag in der Zugriffskontrollliste des übergeordneten Verzeichnisses für die Projektarchive hat (Rechtsklick Verzeichnis | Eigenschaften | Sicherheit), der dem Konto vollen Zugriff auf das Verzeichnis gestattet. Andernfalls können Anwender keine Änderungen zum Projektarchiv übertragen.

Selbst wenn Apache als „Lokales System“ läuft, brauchen Sie immer noch einen solchen Eintrag (der in diesem Fall das SYSTEM-Konto sein wird).

Wenn Apache diese Berechtigung nicht gesetzt hat, werden Ihre Benutzer „Zugriff verweigert“-Fehlermeldungen erhalten, die im Apache-Fehlerlog als Fehler 500 erscheinen.

3.7.3. Subversion installieren

1. Laden Sie die neuste Version von Subversion für Apache unter Win32 herunter. Stellen Sie sicher, dass sie die richtige Version für Ihren Apachen wählen, da Sie sonst seltsame Fehlermeldungen beim Neustart erhalten. Für Apache Version 2.2.x gehen Sie zu <http://subversion.tigris.org/servlets/ProjectDocumentList?folderID=8100>.
2. Starten Sie den Subversion-Installer und folgen Sie den Anweisungen. Wenn der Subversion-Installer bemerkt, dass Sie Apache installiert haben, haben Sie es fast geschafft. Wenn er keinen Apache-Server finden konnte, müssen Sie einige zusätzliche Schritte durchführen.
- 3.

Gehen Sie mit dem Windows Explorer ins Installationsverzeichnis von Subversion (für gewöhnlich `c:\Programme\Subversion`) und finden Sie die Dateien `/httpd/mod_dav_svn.so` und `mod_authz_svn.so`. Kopieren Sie diese Dateien ins Apache-Modulverzeichnis (für gewöhnlich `c:\Programme\apache group\apache2\modules`).

4. Kopieren Sie die Dateien `/bin/libdb*.dll` und `/bin/intl3_svn.dll` aus dem Subversion-Installationsverzeichnis in das Apache `/bin` Verzeichnis.
5. Bearbeiten Sie die Konfigurationsdatei von Apache (für gewöhnlich `C:\Programme\Apache Group\Apache2\conf\httpd.conf`) mit einem Texteditor wie Notepad und nehmen Sie die folgenden Änderungen vor:

Entfernen Sie den Kommentar (d.h. das '#'-Zeichen) in den folgenden Zeilen:

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

Fügen Sie die folgenden zwei Zeilen an das Ende des LoadModule-Abschnittes ein.

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

3.7.4. Konfiguration

Jetzt haben Sie Apache und Subversion eingerichtet, aber Apache weiß noch nicht, wie mit Subversion-Clients wie TortoiseSVN umzugehen ist. Um Apache beizubringen, welche URL für Subversion-

Projektarchive benutzt werden soll, müssen Sie die Apache-Konfigurationsdatei (für gewöhnlich platziert in `c:\Programme\apache group\apache2\conf\httpd.conf`) mit einem beliebigen Texteditor (z.B. Notepad) bearbeiten:

1. Am Ende der Konfigurationsdatei fügen Sie die folgenden Zeilen hinzu:

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

Dies konfiguriert Apache so, dass alle Ihre Subversion-Projektarchive physikalisch unterhalb von `D:\SVN` liegen. Die Projektarchive werden der Außenwelt unter folgender URL zu Verfügung gestellt: `http://MyServer/svn/`. Der Zugriff ist auf bekannte Benutzer/Passwörter beschränkt, die in der `passwd`-Datei aufgelistet sind.

2. Um die `passwd`-Datei zu erstellen, öffnen Sie den Kommandozeilenprompt (DOS-Box) noch einmal, wechseln in den `apache2` Ordner (für gewöhnlich `c:\Programme\apache group\apache2`) und erstellen Sie die Datei, indem Sie

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

eintippen. Dies wird eine Datei mit dem Namen `passwd` erstellen, die für die Anmeldung verwendet wird. Zusätzliche Benutzer können mit

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

hinzugefügt werden.

3. Starten Sie den Apache-Dienst noch einmal neu.
4. Öffnen Sie mit Ihrem Web-Browser die URL `http://MeinServer/svn/MeinNeuesProjektarchiv` (wobei `MeinNeuesProjektarchiv` der Name des Subversion-Projektarchivs ist, das Sie zuvor erstellt haben). Wenn alles gut gegangen ist, sollten Sie nach einem Benutzernamen und Passwort gefragt werden und können dann den Inhalt Ihres Projektarchivs sehen.

Eine kurze Erklärung dessen was Sie gerade eingegeben haben:

Eigenschaft	Erklärung
<code><Location /svn></code>	bedeutet, dass die Projektarchive unter der URL <code>http://MyServer/svn/</code> aufrufbar sind.
<code>DAV svn</code>	teilt Apache mit, welches Modul (in diesem Fall das Subversion Modul), für diese URL zuständig ist.
<code>SVNListParentPath on</code>	Für Subversion 1.3 und neuer ermöglicht diese Direktive, die verfügbaren Projektarchive unter <code>SVNParentPath</code> aufzulisten.
<code>SVNParentPath D:\SVN</code>	teilt Apache mit, dass die Projektarchive unterhalb von <code>D:\SVN</code> liegen.
<code>SVNIndexXSLT "/svnindex.xsl"</code>	XML Stylesheet, mit dem die Standardansicht des Projektarchivs im Web-Browser formatiert wird.

Eigenschaft	Erklärung
AuthType Basic	zur Anmeldung wird ein einfaches Verfahren mit Name / Passwort verwendet.
AuthName "Subversion repositories"	wird als Information an das Programm (z.B. Web-Browser, TortoiseSVN) weitergereicht, mit dem sich der Benutzer am Projektarchiv anmeldet.
AuthUserFile passwd	gibt die Passwortdatei für die Anmeldung an.
AuthzSVNAccessFile	Name der Datei in der die Zugriffsrechte <i>innerhalb</i> der Subversion Projektarchive festgelegt sind.
Require valid-user	legt fest, dass nur Personen, die sich korrekt angemeldet haben, auf die URL zugreifen dürfen.

Tabelle 3.1. Apache httpd.conf Einstellungen

Aber das ist nur ein Beispiel. Der Apache Web-Server bietet Ihnen viele weitere Möglichkeiten.

- Wenn Sie möchten, dass Ihr Projektarchiv für jedermann Lesezugriff bietet, aber der Schreibzugriff bestimmten Benutzern vorbehalten bleibt, können Sie die Zeile

```
Require valid-user
```

in

```
<LimitExcept GET PROPFIND OPTIONS REPORT>
Require valid-user
</LimitExcept>
```

ändern.

- Mit einer passwd Datei können Sie den Zugriff auf alle Ihre Projektarchive zentral steuern. Wenn Sie genau einstellen wollen, welche Benutzer / Gruppen welche Berechtigungen in Ihren Projektarchiven haben sollen, entfernen Sie das Kommentarzeichen vor

```
#AuthzSVNAccessFile svnaccessfile
```

und legen Sie die Subversion Zugriffsdatei `svnaccessfile` an. Apache stellt sicher, dass nur angemeldete Benutzer auf Ihre `/svn` Location zugreifen können und wird dann den Benutzernamen an das Subversion `AuthzSVNAccessFile` Modul weiterreichen, das die Benutzerrechte fein gegliedert anhand der Regeln in der Subversion Zugriffsdatei steuert. Beachten Sie, dass die Pfade entweder als `Projektarchiv:Pfad` oder einfach als `Pfad` angegeben werden. Wenn Sie kein Projektarchiv angeben, wird die Regel auf alle Projektarchive unterhalb `SVNParentPath` angewendet. Das Format der von `mod_authz_svn` verwendeten Berechtigungsdatei ist in [Abschnitt 3.7.6, „Pfadbasierte Autorisierung“](#) beschrieben.

- Um die Ansicht des Projektarchivs im Web-Browser 'schöner' zu formatieren, kommentieren Sie die Zeile

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

aus und kopieren die Dateien `svnindex.xsl`, `svnindex.css` und `menucheckout.ico` in das Document Root Verzeichnis des Apache Servers (normalerweise `C:/Programme/Apache Group/Apache2/htdocs`). Dieses Verzeichnis wird mit der `DocumentRoot` Anweisung in der Apache Konfigurationsdatei eingestellt.

Sie können diese drei Dateien direkt aus unserem Projektarchiv <http://tortoisesvn.googlecode.com/svn/trunk/contrib/svnindex> herunterladen. ([Abschnitt 3, „TortoiseSVN ist frei!“](#) erklärt, wie man auf das TortoiseSVN Projektarchiv zugreift).

Die XSL Datei aus dem TortoiseSVN Projektarchiv enthält einen netten Kniff. Wenn Sie das Projektarchiv mit ihrem Web-Browser betrachten, wird neben jedem Ordner Ihres Projektarchivs ein Symbol angezeigt. Wenn Sie auf dieses Symbol klicken, wird sofort der Auschecken-Dialog von TortoiseSVN für die entsprechende URL aufgerufen.

3.7.5. Mehrere Projektarchive

Wenn Sie die `SVNParentPath`-Direktive benutzt haben, brauchen Sie die Apache-Konfigurationsdatei nicht jedes Mal zu ändern, wenn Sie ein neues Subversion-Projektarchiv erstellt haben. Erstellen Sie das neue Projektarchiv einfach an der gleichen Stelle wie das erste Projektarchiv und Sie sind fertig! In meiner Firma habe ich über SMB (normaler Windows-Dateizugriff) direkten Zugriff auf diesen speziellen Ordner auf dem Server. Also erstelle ich dort einfach einen neuen Ordner, führe den TortoiseSVN-Befehl `TortoiseSVN → Projektarchiv hier erstellen...` aus und ein neues Projekt hat ein Zuhause...

Wenn Sie Subversion 1.3 oder neuer benutzen, können Sie die `SVNListParentPath` on Direktive verwenden, damit Apache eine Liste aller verfügbaren Projekte erzeugt, falls Sie auf den `SVNParentPath` anstatt auf ein Projektarchiv zugreifen.

3.7.6. Pfadbasierte Autorisierung

Das `mod_authz_svn` Modul ermöglicht fein gegliederte Zugriffsberechtigungen basierend auf Benutzernamen und Pfaden im Projektarchiv. Diese Funktionen stehen seit jeher mit dem Apache Server und seit Subversion 1.3 auch mit `svnserve` zur Verfügung.

Eine Beispieldatei könnte folgendermaßen aussehen:

```
[groups]
Admin = john, kate
Entwickler1 = john, rachel, sally
Entwickler2 = kate, peter, mark
Doku = bob, jane, mike
Training = zak
# Standardzugriffsrechte für alle Projektarchive
# Jeder erhält Lesezugriff, Admins können schreiben,
# Karl Napf wird ausgeschlossen
[/]
* = r
@Admin = rw
karlnapf =
# Die Entwickler erhalten Vollzugriff auf ihre Projektarchive
# Hans Dampf arbeitet bei dem großen Projekt noch mit
[Projekt1:/]
@Entwickler1 = rw
[Projekt2:/]
@Entwickler2 = rw
[Riesenprojekt:/]
@Entwickler1 = rw
@Entwickler2 = rw
hansdampf = rw
#Die Doku Gruppe erhält Vollzugriff auf die doc Unterverzeichnisse
[/trunk/doc]
@Doku = rw
# Anfänger dürfen nur im Training Projektarchiv wüten
[TrainingRepos:/]
@Training = rw
```

Beachten Sie, dass die Überprüfung aller Pfade, besonders wenn das Log angezeigt wird, eine teure Operation sein kann. Der Server prüft jeden geänderten Pfad in jeder Revision auf Lesbarkeit, was bei Revisionen, die eine große Anzahl Dateien betreffen, sehr Zeitaufwändig sein kann.

Anmeldung und Zugriffsberechtigung (Autorisierung) sind separate Prozesse. Damit ein Anwender auf einen Pfad im Projektarchiv zugreifen kann, muss er *beide* Bedingungen erfüllen. Er muss angemeldet sein und die Berechtigung zum Zugriff auf den gewünschten Pfad besitzen.

3.7.7. Anmeldung an einer Windows Domäne

Wie Sie bestimmt bemerkt haben, benötigen Sie für jeden Anwender einen eigenen Eintrag in der `passwd` Datei. Falls Sie aus Sicherheitsgründen verlangen, dass die Benutzerpasswörter regelmäßig geändert werden, müssen Sie das jedes Mal von Hand machen.

Falls Sie auf das Projektarchiv in einem Windows Netzwerk mit einem Domänencontroller zugreifen, gibt es eine einfache Lösung: `mod_auth_sspi!`

Das Original SSPI Modul wurde von Syneapps inklusive Quellcode zur Verfügung gestellt. Das Modul wurde jedoch nicht mehr weiterentwickelt. Aber keine Angst, die Open-Source Gemeinde hat sich des Moduls angenommen und es weiterentwickelt. Das Projekt hat ein neues Zuhause auf [SourceForge](http://sourceforge.net/projects/mod-auth-sspi/) [http://sourceforge.net/projects/mod-auth-sspi/].

- Laden Sie das Modul, das zu Ihrer Apache-Version passt herunter und kopieren Sie die Datei `mod_auth_sspi.so` in den Apache `modules` Ordner.

- Bearbeiten Sie die Apache Konfigurationsdatei. Fügen Sie dort die Zeile

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

in den `LoadModule` Abschnitt ein. Stellen Sie sicher, dass Sie diesen Eintrag *vor* der Zeile

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

einfügen.

- Damit die Subversion `Location` dieses Anmeldeverfahren verwendet, müssen Sie die Zeile

```
AuthType Basic
```

in

```
AuthType SSPI
```

ändern. Weiterhin müssen Sie die Zeilen:

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

in den `<Location /svn>` Abschnitt einfügen. Wenn Sie keinen Domänencontroller haben, lassen Sie den Eintrag für den Namen `<domaincontroller>` unverändert.

Beachten Sie, dass Sie bei Verwendung von SSPI keine `AuthUserFile` Zeile mehr benötigen, um eine Passwortdatei zu definieren. Apache wird Ihren Namen und Ihr Passwort gegen den Windows Domänencontroller verifizieren. Sie müssen die Benutzerliste in `svnaccessfile` anpassen, um Anmeldungen der Form `DOMÄNE\username` ebenfalls zuzulassen.



Wichtig

Die SSPI Anmeldung wird nur für Verbindungen aktiviert, die per SSL gesichert sind (https). Wenn Sie nur normale http Verbindungen zu Ihrem Server aufbauen, steht SSPI nicht zur Verfügung.

Eine Anleitung zum Einrichten von SSL auf Ihrem Server findet sich in [Abschnitt 3.7.9](#), „Den Server mit SSL sichern“.



Tipp

Subversion AuthzSVNAccessFile Dateien berücksichtigen bei Benutzernamen die Groß- / Kleinschreibung (JUser unterscheidet sich von juser).

In der Microsoft Welt wird die Schreibweise bei Windows Domänen und Benutzernamen ignoriert. Trotzdem legen viele Administratoren gerne Benutzerkonten wie JUser an.

Dieser Unterschied kann schmerzhaft sein, wenn Sie SSPI Anmeldung unter Windows verwenden und der Benutzername in der gleichen Schreibweise an Subversion weitergereicht wird, wie der Anwender ihn eingegeben hat, da JUser und juser für Subversion verschiedene Personen sind. Der Internet Explorer reicht den Anmeldenamen oft automatisch in der Schreibweise, in der das Konto angelegt wurde, an Apache weiter.

Das Ergebnis davon ist, dass Sie für jeden Anwender mindestens zwei Einträge in Ihrer AuthzSVNAccessFile Datei benötigen. Einen Eintrag in Kleinbuchstaben und einen Eintrag in der Schreibweise wie das Konto angelegt wurde. Weiterhin müssen Sie Ihre Anwender dahingehend trainieren, dass sie sich immer in Kleinbuchstaben anmelden, wenn sie einen Subversion Client wie TortoiseSVN benutzen.

Die Fehler- und Zugriffs-Logdateien von Apache sind Ihre besten Freunde, wenn es darum geht, Probleme dieser Art zu lokalisieren und zu beseitigen, da Sie Ihnen anzeigen, in welcher Schreibweise ein Anmelde-name an das Subversion AuthzSVNAccessFile Modul weitergereicht wurde. Sie müssen eventuell etwas mit dem exakten Format der Anmeldenamen in svnaccessfile experimentieren (z.B. DOMÄNE\user und DOMÄNE//user), um alles ans Laufen zu bekommen.

3.7.8. Mehrere Anmeldequellen

Es ist auch möglich, mehr als eine Anmeldequelle für Ihr Subversion Projektarchiv zu haben. Um dies zu erreichen, müssen Sie jeden Anmeldetyp als unmaßgeblich („non-authoritative,“) festlegen, so dass Apache mehrere Quellen auf eine korrekte Name/Passwort Kombination überprüft.“

Ein übliches Szenario ist es, sowohl die Windows Domänenanmeldung als auch eine passwd Datei zu verwenden, so dass Sie Benutzern, die kein Domänenkonto besitzen, Zugriff auf das Projektarchiv gewähren können.

- Um sowohl die Windows Domänenanmeldung als auch Anmeldung per Passwortdatei zu ermöglichen, fügen Sie die folgenden Einträge innerhalb des <Location> Blocks Ihrer Apache-Konfiguration ein:

```
AuthBasicAuthoritative Off
SSPIAuthoritative Off
```

Hier ist ein Beispiel einer vollständigen Apache Konfiguration für gemeinsamen Windows Domänenzugriff und passwd Dateianmeldung:

```
<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. Den Server mit SSL sichern

Obwohl Apache 2.2.x OpenSSL unterstützt, wird diese Option standardmäßig nicht genutzt und muss manuell aktiviert werden.

1. Kommentieren Sie die folgenden Zeilen in der Apache Konfigurationsdatei aus:

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

und am Ende:

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

Dann ändern Sie die Zeile (in einer Zeile):

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

in

```
SSLMutex default
```

2. Als nächstes benötigen Sie ein SSL Zertifikat. Dazu starten Sie eine Kommandozeile (DOS-Box) und wechseln in den Apache Ordner (z.B. C:\Programme\apache_group\apache2) und geben den folgenden Befehl ein:

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

Sie werden nach einer Passphrase gefragt. Bitte nutzen Sie keine einfachen Wörter sondern ganze Sätze, z.B. ein Teil eines Gedichts. Je länger die Passphrase ist, desto besser. Außerdem müssen Sie die URL Ihres Servers angeben. Alle anderen Angaben sind optional. Sie sollten Sie aber trotzdem ausfüllen.

Normalerweise wird die `privkey.pem` Datei automatisch generiert. Wenn nicht, müssen Sie sie mit folgendem Befehl selbst erzeugen:

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

Danach geben Sie folgende Befehle, jeweils auf einer Zeile ein:

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

und

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

und danach

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

Dies wird ein Zertifikat erstellen, das in 4000 Tagen verfällt. Abschließend geben Sie:

```
bin\openssl x509 -in conf\server.crt -out conf\server.der.crt  
-outform DER
```

ein. Diese Befehle legen folgende Dateien im Apache conf Ordner an: server.der.crt, server.csr, server.key, .rnd, privkey.pem, server.cert.

3. Starten Sie den Apache Dienst neu.

4. Rufen Sie Ihren Web-Browser mit der URL `https://servername/svn/project` auf.



SSL und Internet Explorer

Wenn Sie Ihren Server mit SSL sichern und sich über eine Windows Domäne anmelden, werden Sie feststellen, dass die Anzeige des Projektarchivs im Internet Explorer nicht mehr funktioniert. Das liegt daran, dass sich der Internet Explorer nicht mehr anmelden kann. Andere Web-Browser haben dieses Problem nicht und sowohl TortoiseSVN als auch andere Subversion Clients können sich problemlos anmelden.

Wenn Sie trotzdem IE verwenden wollen, um Projektarchive zu betrachten, können Sie entweder:

- Einen separaten `<Location /path>` Block in der Apache Konfigurationsdatei definieren und den Eintrag `SSPIBasicPreferred On` hinzufügen. Dies ermöglicht IE die Anmeldung wieder, aber andere Web-Browser und Subversion können sich an dieser Location nicht anmelden.
- Zusätzlich das Betrachten des Projektarchivs ohne Verschlüsselung (ohne SSL) anbieten. Seltsamerweise hat IE kein Problem mit der Anmeldung, wenn die Übertragung nicht verschlüsselt erfolgt.
- In der SSL „Standardkonfiguration“ findet sich häufig der folgende Eintrag beim virtuellen SSL Host:

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

Es gibt (gab?) gute Gründe für diesen Eintrag. Siehe [ModSSL FAQ](http://www.modssl.org/docs/2.8/ssl_faq.html#ToC49) [http://www.modssl.org/docs/2.8/ssl_faq.html#ToC49]. Wenn Sie allerdings das NTLM Anmeldeverfahren nutzen wollen müssen Sie `keepalive` verwenden. Wenn Sie den gesamten `SetEnvIf` Block auskommentieren, sollten Sie sich mit dem IE per Windows Anmeldung über SSL und dem eingebundenen `mod_auth_sspi` Modul am Apache anmelden können.



SSL Zugriff erzwingen

Wenn Sie SSL eingerichtet haben, um Ihr Projektarchiv abzusichern, möchten Sie vielleicht den normalen, unverschlüsselten `http://` Zugriff sperren und nur `https://` zulassen. Dafür müssen Sie zusätzlich die Anweisung `SSLRequireSSL` in den Subversion `<Location>` Block einfügen.

Ein Beispiel `<Location>` Block könnte folgendermaßen aussehen:

```
<Location /svn>
  DAV svn
  SVNParentPath D:\SVN
  SSLRequireSSL
  AuthType Basic
  AuthName "Subversion Projektarchive"
  AuthUserFile passwd
  #AuthzSVNAccessFile svnaccessfile
  Require valid-user
</Location>
```

3.7.10. Client-Zertifikate mit virtuellen SSL Hosts verwenden

Danke an Nigel Green für diesen Beitrag an die TortoiseSVN Mailingliste!

In manchen Serverkonfigurationen müssen Sie einen einzelnen Server mit zwei virtuellen Hosts einrichten: Den ersten für öffentlichen Web-Zugriff ohne Zertifikat, den zweiten, auf dem ein Subversion Server läuft, durch ein für den Client erforderliches Zertifikat abgesichert.

Durch das Hinzufügen einer `SSLVerifyClient Optional` Anweisung in die *per-server* Sektion der Apache Konfiguration (d.h. außerhalb aller `VirtualHost` und `Directory` Blöcke) zwingen Sie Apache, ein Client-Zertifikat während des ersten SSL Handshake anzufordern. Aufgrund eines Fehlers in `mod_ssl` ist es wichtig, dass das Zertifikat zu diesem Zeitpunkt angefordert wird, da es nicht funktioniert, wenn die SSL Verbindung neu ausgehandelt wird.

Die Lösung besteht darin, dem Virtual Host Verzeichnis, das sie für Subversion absichern wollen, die folgende Anweisung hinzuzufügen:

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

Diese Anweisung ermöglicht nur dann den Zugriff auf das Verzeichnis, falls ein Client-Zertifikat empfangen und erfolgreich verifiziert wurde.

Zusammengefasst sehen die relevanten Zeilen der Apache Konfiguration folgendermaßen aus:

```
SSLVerifyClient Optional

### Virtual Host Konfiguration für den ÖFFENTLICHEN Host
### (ohne dass ein Zertifikat benötigt wird)

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

### Virtual Host Konfiguration für SUBVERSION
### (ein Client-Zertifikat wird benötigt)
<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>
```

Kapitel 4. Anleitung zum täglichen Gebrauch

Dieses Kapitel beschreibt den täglichen Umgang mit TortoiseSVN. Es ist *nicht* als Einführung in Versionskontrollsysteme gedacht, und auch nicht als Einführung in Subversion (SVN). Dies ist mehr ein Nachschlagewerk wenn Sie nicht mehr genau wissen wie eine Funktion ausgeführt wird, sie aber zumindest wissen was Sie tun wollen.

Falls Sie eine Anleitung für Versionskontrolle mit Subversion benötigen, empfehlen wir das fantastische Buch *Versionskontrolle mit Subversion* [<http://svnbook.red-bean.com/>].

An diesem Dokument wird ständig gearbeitet, wie auch an TortoiseSVN und Subversion immer weitergearbeitet wird. Falls Sie Fehler feststellen, melden Sie diese an die Mailing-Liste damit wir die Dokumentation aktualisieren können. Einige der Screenshots in diesem Dokument zeigen wahrscheinlich nicht den aktuellsten Stand der Software. Seien Sie bitte nachsichtig. Wir arbeiten an TortoiseSVN und der Dokumentation während unserer Freizeit.

Um das Beste aus dem Daily Use Guide herauszuholen:

- Sie haben TortoiseSVN bereits installiert.
- Sie kennen sich mit Versionskontrollsystemen aus.
- Sie kennen die Grundlagen von Subversion.
- Sie haben einen Server installiert und/oder haben Zugriff auf ein Subversion Projektarchiv.

4.1. Vorbereitungen

4.1.1. Überlagerte Symbole

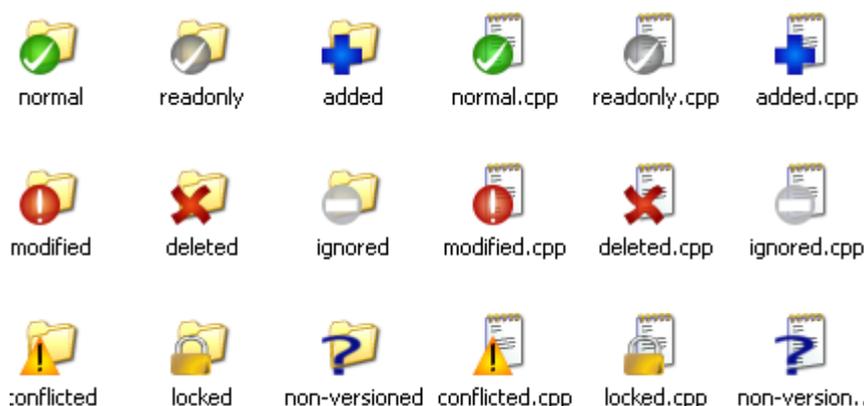


Abbildung 4.1. Explorer mit überlagerten Symbolen

Eine der auffälligsten Eigenschaften von TortoiseSVN sind die überlagerten Symbole, die auf Dateien und Ordnern in Ihrer Arbeitskopie erscheinen. Sie zeigen Ihnen auf einen Blick an, welche Objekte verändert wurden. Lesen Sie in **Abschnitt 4.7.1, „Überlagerte Symbole“** nach, was die einzelnen Symbole bedeuten.

4.1.2. Kontextmenüs

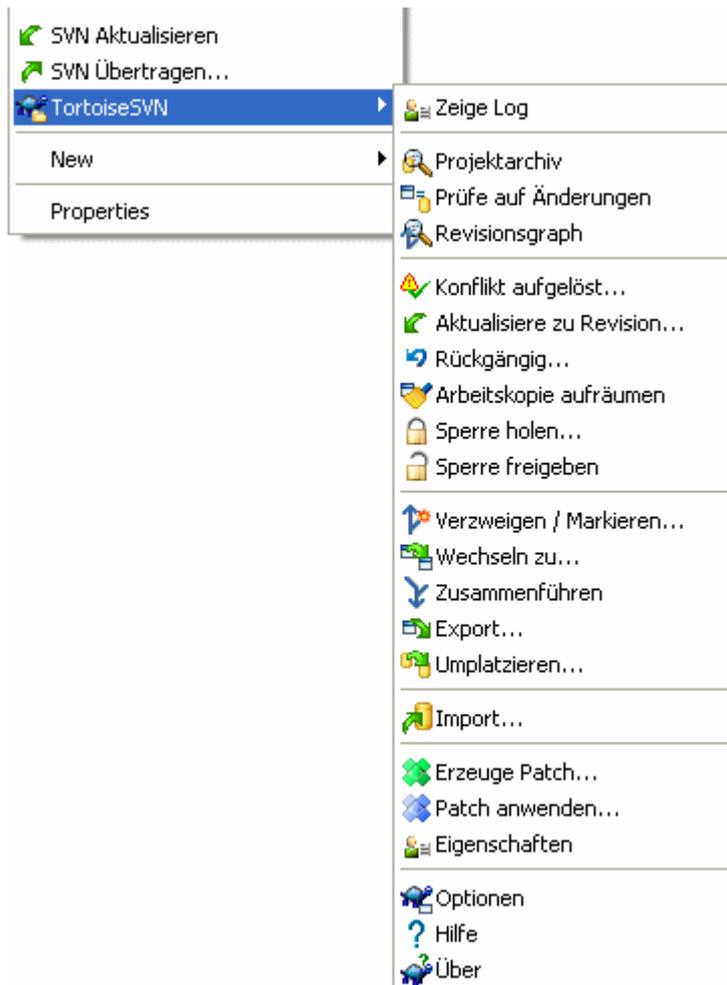


Abbildung 4.2. Kontextmenü für einen Ordner unter Versionskontrolle

Alle Befehle von TortoiseSVN werden über das Kontextmenü des Windows Explorers aufgerufen. Die meisten sind direkt sichtbar, wenn Sie einen Rechtsklick auf eine Datei oder Ordner machen. Welche Befehle angezeigt werden, hängt davon ab ob das angeklickte Objekt oder sein übergeordneter Ordner unter Versionskontrolle stehen oder nicht. Das TortoiseSVN Menü wird auch im Dateimenü des Explorers angezeigt.



Tipp

Manche, selten genutzte Befehle stehen nur über das erweiterte Kontextmenü zur Verfügung. Um das erweiterte Kontextmenü aufzurufen, halten Sie die **Umsch**-Taste gedrückt, während Sie einen Rechtsklick machen.

In manchen Fällen werden Sie mehrere TortoiseSVN Einträge sehen. Das ist kein Fehler!

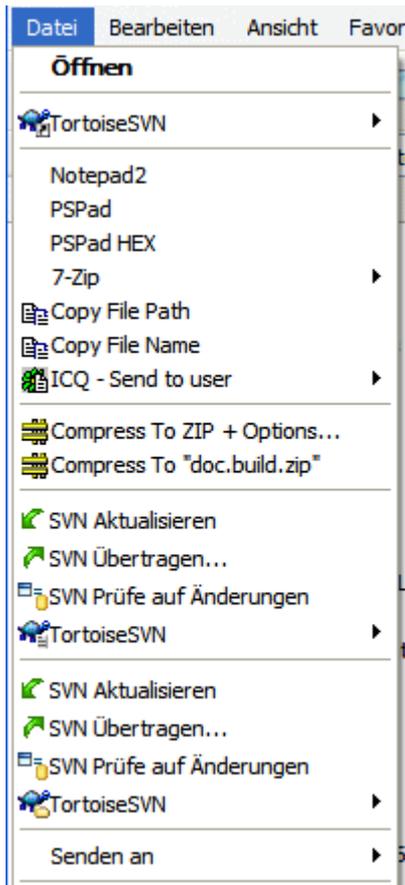


Abbildung 4.3. Explorer Kontextmenü für Verknüpfungen in einem versionierten Ordner

In Diesem Beispiel sehen Sie eine nicht versionierte Verknüpfung in einem versionierten Ordner, und das Dateimenü des Explorers zeigt *drei* Einträge für TortoiseSVN an. Einer steht für den Ordner, einer für die Verknüpfung und einer für das Objekt auf das die Verknüpfung zeigt. Damit Sie die drei Menüs unterscheiden können, besitzen die Symbole einen Indikator in der unteren rechten Ecke, der anzeigt, ob es sich um eine Datei, einen Ordner, eine Verknüpfung oder eine Mehrfachauswahl handelt.

Wenn Sie Windows 2000 benutzen, werden Sie feststellen, dass die Kontextmenüs nur Text und keine Symbole enthalten. Uns ist bekannt, dass ältere Versionen von TortoiseSVN unter Windows 2000 die Symbole angezeigt haben, aber Microsoft hat die Behandlung der Menüsymbole in Vista geändert. Daran mussten wir unsere Darstellungsroutinen anpassen, wodurch leider die Symbole unter Windows 2000 entfallen sind.

4.1.3. Ziehen und Ablegen

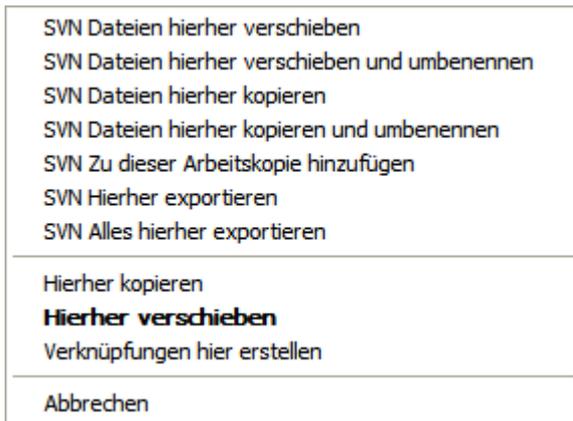


Abbildung 4.4. Rechts-Ziehen-Menü für einen Ordner unter Versionskontrolle

Andere Befehle sind als Ziehen und Ablegen Funktion vorhanden. Wenn Sie Dateien oder Ordner innerhalb einer Arbeitskopie Rechts-Ziehen oder wenn Sie eine nicht versionskontrollierte Datei in eine Arbeitskopie Rechts-Ziehen.

4.1.4. Tastaturkürzel

Einige allgemeine Aktionen haben wohlbekannte Windowstastenkürzel, werden aber nicht auf Schaltflächen oder in Menüs angezeigt. Wenn Ihnen einige wichtige Funktionen, wie Aktualisieren der Anzeige, nicht klar sind, schauen Sie hier nach.

F1
Hilfe, was sonst?

F5
Aktualisiert die Anzeige. Dies ist vielleicht der wichtigste Eintastenbefehl. Im Explorer wird diese Funktion die überlagerten Symbole aktualisieren. Im Übertragen-Dialog wird sie die Arbeitskopie erneut durchsuchen, um Ihnen die geänderten Dateien anzuzeigen. Im Logdialog wird sie das Projektarchiv erneut kontaktieren, um die neuesten Änderungen anzuzeigen.

Strg+A
Wähle alles. Diese Befehl kann zum Beispiel verwendet werden, um eine Fehlermeldung in eine E-Mail zu kopieren, oder um eine Liste der geänderten Dateien zu kopieren oder ...

Strg+C
Kopiert den markierten Text

4.1.5. Anmeldung

Wenn das Projektarchiv, auf das Sie zugreifen wollen, passwortgeschützt ist, erscheint folgender Anmeldedialog.

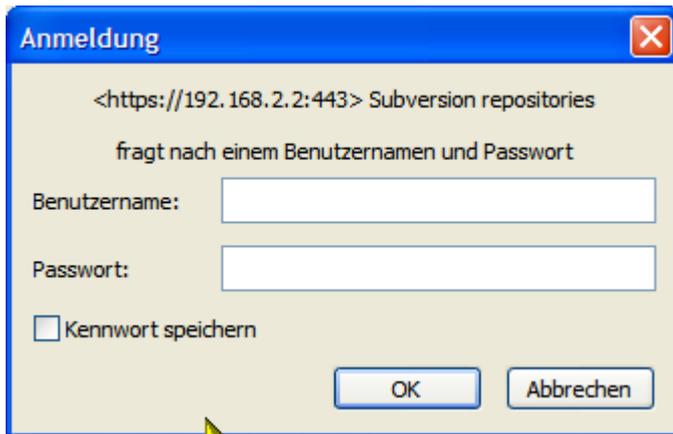


Abbildung 4.5. Anmeldedialog

Geben Sie Ihren Benutzernamen und Ihr Passwort ein. Die Option bewirkt, dass TortoiseSVN die Anmeldeinformationen in einem von drei Unterordnern des Subversion Konfigurationsverzeichnis `%APPDATA%\Subversion\auth` ablegt:

- `svn.simple` enthält die Daten für das einfache Anmeldeverfahren (Name/Passwort).
- `svn.ssl.server` enthält SSL Server Zertifikate.
- `svn.username` enthält die Daten für Anmeldung per Benutzernamen (ohne Passwort).

Wenn Sie die Anmeldedaten für *alle* Server löschen wollen, können Sie dies von der **Gespeicherte Daten** Seite der TortoiseSVN Optionen aus tun. Die Schaltfläche löscht sämtliche Anmeldedaten aus den Subversion `auth` Verzeichnissen sowie die durch ältere Versionen von TortoiseSVN in der Registrierung abgelegten Anmeldedaten. Weitere Informationen stehen in [Abschnitt 4.30.6, „Gespeicherte Daten“](#).

Wenn Sie möchten, dass Ihre Subversion-Anmeldeinformationen beim Abmelden oder Herunterfahren von Windows gelöscht werden, erstellen Sie ein Skript, das das Verzeichnis `%APPDATA%\Subversion\auth` löscht. z.B.

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

Eine Erklärung, wie man solche Skripte installiert finden Sie unter [windows-help-central.com](http://www.windows-help-central.com/windows-shutdown-script.html) [<http://www.windows-help-central.com/windows-shutdown-script.html>].

Für weitere Informationen über Serverkonfiguration und Zugriffskontrolle lesen Sie bitte [Abschnitt 3.5, „Zugriff auf das Projektarchiv“](#).

4.1.6. Fenster maximieren

Viele der Dialoge von TortoiseSVN zeigen eine große Menge an Informationen an. Manchmal ist es sinnvoll, anstatt das Fenster auf Vollbild zu vergrößern, nur die Breite oder die Höhe zu maximieren. Zu diesem Zweck stehen Ihnen Abkürzungen auf der Maximieren Schaltfläche zur Verfügung. Klicken Sie mit der mittleren Maustaste, wird vertikal maximiert, mit der rechten Maustaste horizontal.

4.2. Daten in ein Projektarchiv importieren

4.2.1. Importieren

Wenn Sie in ein bereits existierendes Projektarchiv importieren, das schon einige Projekte enthält, ist dessen Struktur bereits festgelegt. Wenn Sie Daten in ein neues Projektarchiv importieren, sollten Sie sich vorher darüber Gedanken machen, wie Sie es organisieren. Lesen Sie [Abschnitt 3.1.5, „Struktur des Projektarchivs“](#) für weitere Hinweise.

Dieser Abschnitt beschreibt den Import-Befehl von Subversion, der dazu gedacht ist, eine Verzeichnisstruktur in ein Projektarchiv zu importieren. Obwohl er für diese Aufgabe geeignet ist, besitzt er doch einige Nachteile:

- Es gibt, abgesehen von den globalen Ignoriermustern, keine Möglichkeit Dateien oder Ordner einzuschließen.
- Der importierte Ordner wird nicht zu einer Arbeitskopie. Sie müssen eine Arbeitskopie aus dem Projektarchiv auschecken.
- Es kann leicht passieren, dass Sie die Daten in einen falschen Ordner im Projektarchiv importieren.

Aus diesen Gründen empfehlen wir, dass Sie den Import-Befehl gar nicht nutzen sondern die in [Abschnitt 4.2.2, „Import an Ort und Stelle“](#) beschriebene zweistufige Methode. Aber da sie schon diesen Abschnitt lesen, folgt nun eine kurze Einführung ...

Bevor Sie ein Projekt in das Projektarchiv importieren sollten Sie:

1. Alle Dateien entfernen/löschen welche nicht unbedingt für das Projekt notwendig sind (z.B. temporäre Dateien, Dateien die vom Compiler erzeugt werden wie *.obj, kompilierte EXE Dateien, ...)
2. Die Dateien und Ordner optimal anordnen. Obwohl es auch später noch immer möglich ist, die Dateien und Ordner umzubenennen oder zu verschieben, ist es doch empfehlenswert, schon beim Importieren eine saubere Struktur zu haben.

Wählen Sie nun den *übergeordneten* Ordner Ihrer Ordnerstruktur im Windows Explorer und öffnen Sie mit einem Rechtsklick das Kontextmenü. Wählen Sie den Befehl TortoiseSVN → Importieren... worauf der folgende Dialog erscheint:

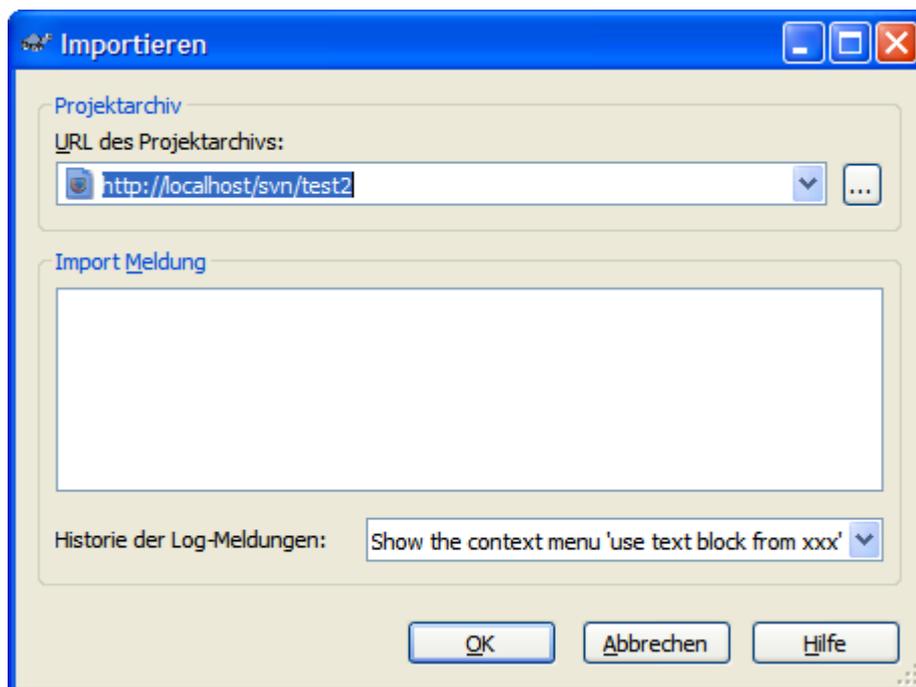


Abbildung 4.6. Der Import-Dialog

In diesem Dialog geben Sie die URL des Projektarchivs ein, in das Sie Ihr Projekt importieren wollen. Es ist sehr wichtig zu verstehen, dass der lokale Ordner, den Sie importieren, nicht im Projektarchiv landet, sondern nur sein Inhalt. Wenn Sie z.B. die folgende Struktur haben:

```
C:\Projekte\Grafik\source
C:\Projekte\Grafik\doku
C:\Projekte\Grafik\bilder
```

und Sie importieren C:\Projekte\Grafik in `http://meinserver.com/svn/trunk` könnten Sie überrascht feststellen, dass Ihre Unterverzeichnisse direkt in `trunk` anstelle eines `Grafik` Unterverzeichnisses landen. Sie müssen das Unterverzeichnis als Teil der URL angeben `http://meinserver.com/svn/trunk/Grafik`. Beachten Sie, dass der Import-Befehl automatisch Unterverzeichnisse im Projektarchiv anlegt, falls diese noch nicht existieren.

Die Importmeldung wird als Logmeldung verwendet.

Standardmäßig werden Dateien und Ordner, die dem globalen Ignoriermuster entsprechen, *nicht* importiert. Um dieses Verhalten zu übergehen, aktivieren Sie die Option *Ignorierte Dateien einschließen*. Lesen Sie in [Abschnitt 4.30.1, „Allgemeine Einstellungen“](#) nach, wie man globale Ignoriermuster einrichtet.

Sobald Sie auf **OK** klicken, beginnt TortoiseSVN die Daten in das Projektarchiv zu importieren. Beachten Sie bitte, dass dadurch Ihr Importverzeichnis *nicht* unter Versionskontrolle gestellt wird! Um eine Arbeitskopie zu erhalten, in der die Daten unter Versionskontrolle sind müssen sie die Daten frisch aus dem Projektarchiv auschecken. Oder sie Lesen weiter, um herauszufinden wie man beim Import eine Arbeitskopie an Ort und Stelle erzeugen kann.

4.2.2. Import an Ort und Stelle

Angenommen, Sie haben bereits ein Projektarchiv und wollen eine neue Ordnerstruktur hinzufügen, folgen Sie diesen Schritten:

1. Erstellen Sie mit Hilfe des Projektarchivbetrachters einen neuen Ordner direkt im Projektarchiv.
2. Checken Sie den neu erstellten Ordner über den zu importierenden Ordner aus. Es wird eine Warnung angezeigt, dass der Zielordner nicht leer ist. Nun haben Sie einen lokal versionierten Ordner mit nicht versioniertem Inhalt.
3. Wählen Sie TortoiseSVN → *Hinzufügen...* auf dem versionierten Ordner, um Objekte zur Versionskontrolle hinzuzufügen. Sie können Dateien hinzufügen oder löschen, die `svn:ignore` Eigenschaft für Ordner setzen und weitere Änderungen vornehmen.
4. Übertragen Sie den obersten Ordner und sie erhalten nun eine versionierte Ordnerstruktur im Projektarchiv sowie eine Arbeitskopie, die aus dem existierenden Ordner heraus angelegt wurde.

4.2.3. Spezielle Dateien

Manchmal ist es notwendig, eine Datei unter Versionskontrolle zu haben, die benutzerspezifische Daten (z.B. absolute Pfade zu Anwendungen) enthält. Das bedeutet Sie haben eine Datei, die von jedem Benutzer verändert werden muss, um sie an seine lokalen Einstellungen anzupassen. Aber eine Datei unter Versionskontrolle würde von jedem Benutzer jeweils wieder zum Projektarchiv übertragen werden und so die Änderungen von anderen Benutzern wieder überschreiben.

In solchen Fällen empfehlen wir die Verwendung von so genannten *Schablonen*. Eine Schablone ist nichts anderes als eine normale Datei, welche entweder einen anderen Dateinamen oder eine andere Dateiendung hat als die Datei, welche schlussendlich verwendet wird.

Als Beispiel sehen Sie sich einmal das Erstellungsskript von TortoiseSVN an. Es ruft eine Datei namens `TortoiseVars.bat` auf, die im Projektarchiv gar nicht existiert! Es existiert aber

die Datei `TortoiseVars.tmpl`, welche die Schablone für die Datei `TortoiseVars.bat` darstellt. Bevor also das Skript ausgeführt werden kann muss jeder Benutzer eine Kopie von `TortoiseVars.tmpl` erstellen und die Kopie in `TortoiseVars.bat` umbenennen. Dann kann die Datei `TortoiseVars.bat` ohne Probleme so verändert werden, dass die absoluten Pfade zu den zur Erstellung von TortoiseSVN notwendigen Programmen mit den lokalen Pfaden übereinstimmen.

Um die Benutzer nicht zu stören, ist die Datei `TortoiseVars.bat` auch in der Liste der ignorierten Dateien eingetragen. Das heißt wir haben die Subversion Eigenschaft `svn:ignored` für diese Datei gesetzt. Damit erscheint diese Datei nicht in jedem Übertragen-Dialog als (noch) nicht versioniert.

4.3. Eine Arbeitskopie auschecken

Um eine Arbeitskopie zu erhalten müssen Sie zunächst die Dateien aus einem Projektarchiv *Auschecken*.

Wählen Sie nun den Ordner im Windows Explorer in dem Sie Ihre Arbeitskopie erstellen wollen und öffnen Sie mit einem Rechtsklick das Kontextmenü. Wählen Sie den Befehl TortoiseSVN → Auschecken... worauf der folgende Dialog erscheint:

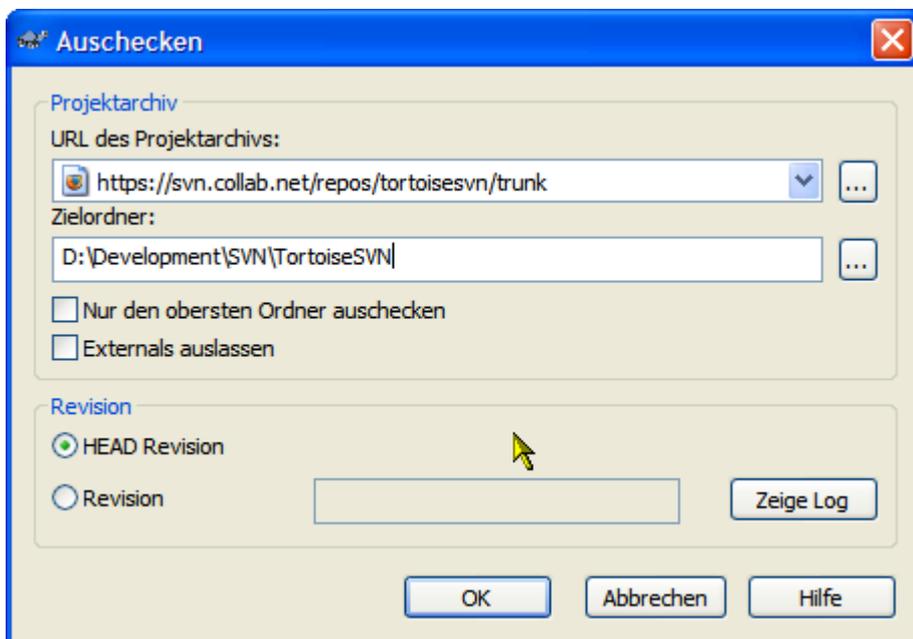


Abbildung 4.7. Der Auschecken-Dialog

Wenn Sie einen Ordnernamen angeben, der noch nicht existiert, wird dieser Ordner im Projektarchiv angelegt.

4.3.1. Rekursionstiefe

Sie können die *Tiefe* der Rekursion beim Auschecken festlegen. Wenn Sie nur einige Bereiche eines großen Quellbaumes wollen, können Sie den obersten Ordner auschecken und danach die gewünschten Ordner rekursiv aktualisieren.

Vollständig rekursiv

Checkt den gesamten Baum rekursiv inklusive aller Dateien und Unterordner aus.

Direkte Kinder, inklusive Ordner

Checkt das angegebene Verzeichnis inklusive aller Dateien und Unterordner aus, füllt die Verzeichnisse aber nicht.

Nur Dateien

Checkt das angegebene Verzeichnis inklusive aller Dateien aus. Es werden keine Unterordner angelegt.

Nur dieses Objekt

Checkt nur das angegebene Verzeichnis aus. Weder Dateien noch Unterordner werden angelegt.

Arbeitskopie

Behält die in der Arbeitskopie angegebene Rekursionstiefe bei. Diese Option wird im Auschecken-Dialog nicht verwendet, ist aber die Vorgabe in allen anderen Dialogen mit einer Angabe der Rekursionstiefe.

Ausschließen

Wird verwendet, um die Tiefe der Arbeitskopie zu reduzieren, nachdem ein Ordner bereits gefüllt wurde. Diese Option steht nur im Aktualisiere zu Revision Dialog zur Verfügung.

Wenn Sie eine dünn besiedelte Arbeitskopie auschecken, indem Sie etwas anderes als vollständig rekursiv für die Auschecktiefe wählen, können Sie später weitere Ordner mittels des Projektarchivbetrachters ([Abschnitt 4.24, „Projektarchivbetrachter“](#)) oder des Auf Änderungen prüfen-Dialoges ([Abschnitt 4.7.3, „Prüfe auf Änderungen“](#)) holen.

Rufen Sie mit einem Rechtsklick auf den ausgecheckten Ordner den Projektarchivbetrachter auf. Suchen Sie den Unterordner, den Sie zu Ihrer Arbeitskopie hinzufügen wollen und wählen Sie **Kontextmenü** → **Aktualisiere zu Revision...** Dieser Menüeintrag ist nur sichtbar, wenn das markierte Objekt noch nicht in der Arbeitskopie vorhanden ist, aber sein Elternobjekt bereits existiert.

Im **Prüfe auf Änderungen**-Dialog klicken Sie zunächst auf **Projektarchiv prüfen**. Der Dialog zeigt alle Dateien, die sich im Projektarchiv befinden, aber von Ihnen noch nicht ausgecheckt wurden, als hinzugefügt. Markieren Sie den Ordner, den Sie zu Ihrer Arbeitskopie hinzufügen möchten mit einem Rechtsklick und wählen Sie **Kontextmenü** → **Aktualisieren**.

Diese Funktion ist sehr nützlich, wenn Sie nur Teile einer großen Projektstruktur auschecken wollen und gleichzeitig die Bequemlichkeit einer einzelnen Arbeitskopie wünschen. Nehmen wir an, Sie haben eine große Verzeichnisstruktur mit den Ordnern `Projekt01` bis `Projekt99` und sie möchten nur die Ordner `Projekt03`, `Projekt25` und `Projekt76/Unterprojekt` auschecken. Dann gehen Sie in folgenden Schritten vor:

1. Checken Sie den Elternordner mit der Tiefe „Nur dieses Objekt“ aus. Damit erhalten Sie ein leeres, übergeordnetes Verzeichnis.
2. Wählen Sie den neuen Ordner und rufen Sie mittels **TortoiseSVN** → **Projektarchiv** den Projektarchivbetrachter auf, um den Inhalt anzuzeigen.
3. Machen Sie einen Rechtsklick auf `Projekt03` und wählen Sie **Kontextmenü** → **Aktualisiere zu Revision...** Behalten Sie die Standardeinstellung bei und Klicken Sie auf **OK**. Damit wird dieser Ordner vollständig befüllt.

Wiederholen Sie diesen Vorgang für `Projekt25`.

4. Navigieren Sie zu `Projekt76/Unterprojekt` und wiederholen Sie die Aktion. Beachten Sie, dass danach der Ordner `Projekt76` bis auf den vollständig gefüllten Unterordner `Unterprojekt` leer ist. Subversion hat für Sie die dazwischen liegenden Ordner angelegt, ohne sie zu füllen.



Die Tiefe der Arbeitskopie ändern

Sobald Sie eine Arbeitskopie mit einer bestimmten Tiefe ausgecheckt haben, können Sie die Tiefe später per **Kontextmenü** → **Aktualisiere zu Revision...** ändern, so dass Sie mehr oder weniger Inhalt erhalten.



Zugriff auf einen älteren Server

Server vor Version 1.5 kennen die Anfrage nach einer bestimmten Tiefe der Arbeitskopie nicht. Deshalb können sie mit solchen Anfragen nicht immer effizient umgehen. Der Befehl wird zwar trotzdem funktionieren, jedoch wird ein älterer Server die vollständigen Daten schicken und es dem Client überlassen, die nicht benötigten Teile auszufiltern, was einige Netzwerklast verursachen kann. Wenn möglich sollten Sie ihren Server auf Version 1.5 aktualisieren.

Wenn das Projekt Verweise auf externe Projekte enthält welche Sie *nicht* mit auschecken möchten, aktivieren Sie die `Externals auslassen` Option.



Wichtig

Falls `Externals auslassen` markiert ist oder falls Sie die Tiefe erhöhen wollen, müssen Sie Aktualisierungen Ihrer Arbeitskopie mittels `TortoiseSVN → Aktualisiere zu Revision...` anstatt `TortoiseSVN → Aktualisieren` durchführen. Die Standardaktualisierung wird alle externen Verweise aktualisieren und die aktuelle Tiefe beibehalten.

Wir empfehlen Ihnen, jeweils nur den `trunk`-Teil des Projektarchivs auszuchecken. Falls Sie den übergeordneten Pfad auschecken, erhalten Sie den kompletten Dateibaum des Projektarchivs! So können Sie sehr schnell Ihre Festplatte füllen, da Sie z.B. für jede Marke in `tags` eine separate Kopie der Daten erhalten!



Exportieren

Manchmal ist es notwendig, eine lokale Kopie ohne die `.svn` Ordner zu haben, zum Beispiel um eine Zip-Datei mit dem Sourcecode zu erstellen. Bitte lesen sie dazu [Abschnitt 4.26, „Eine Arbeitskopie exportieren“](#).

4.4. Ihre Änderungen ins Projektarchiv übertragen

Änderungen an Dateien im Projektarchiv abzuspeichern wird auch *Übertragen* der Änderungen genannt. Vor dem Übertragen jedoch sollten Sie sicherstellen, dass Ihre Arbeitskopie auch auf dem neuesten Stand ist. Sie können entweder den Befehl `TortoiseSVN → Aktualisieren...` sofort ausführen oder zunächst mittels `TortoiseSVN → Prüfe auf Änderungen`, welche Dateien sich lokal oder auf dem Server geändert haben.

4.4.1. Der Übertragen-Dialog

Wenn Ihre Arbeitskopie auf dem neuesten Stand ist und keine Konflikte vorhanden sind, können Sie Ihre Änderungen übertragen. Wählen Sie die Dateien/Ordner aus die Sie übertragen wollen und wählen Sie den Befehl `TortoiseSVN → Übertragen...`

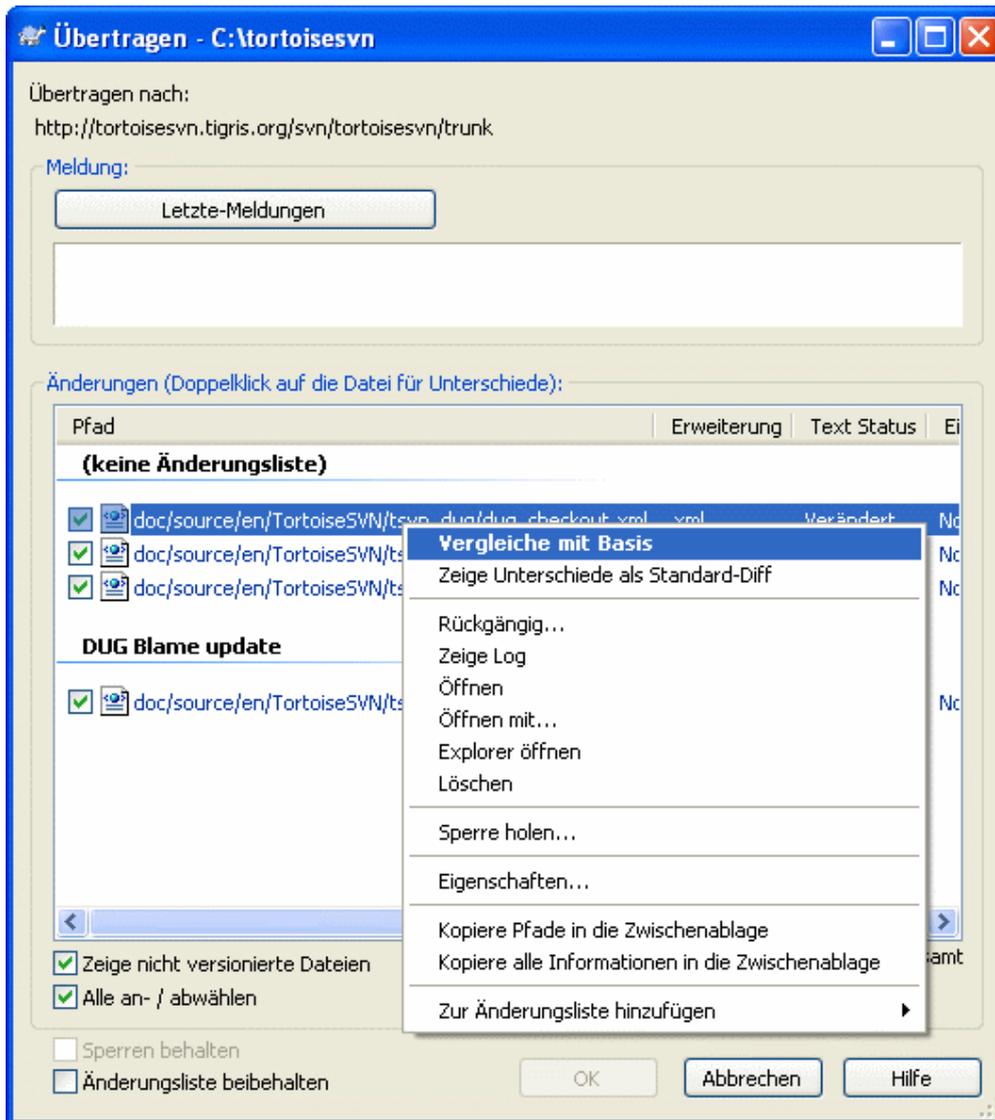


Abbildung 4.8. Der Übertragen-Dialog

Der Dialog zeigt alle geänderten Dateien einschließlich hinzugefügter, gelöschter oder nicht versionierter Dateien an. Wenn Sie bestimmte Dateien nicht übertragen wollen, wählen Sie diese einfach ab. Dateien, die noch nicht unter Versionskontrolle stehen, können Sie durch Markieren vor dem Übertragen hinzufügen.

Objekte, die zu einem anderen Pfad im Projektarchiv gewechselt wurden, werden durch ein (s) gekennzeichnet. Vielleicht haben Sie während der Entwicklung etwas auf einen Zweig umgeschaltet und vergessen, zurück zu trunk zu wechseln. Dies ist Ihr Warnzeichen!



Dateien oder Ordner übertragen?

Wenn Sie Dateien übertragen zeigt der Dialog *nur* die gewählten Dateien an. Wenn Sie Ordner übertragen, werden die geänderten Dateien automatisch selektiert. Übertragen eines Ordners bedeutet *nicht*, dass jede Datei in diesem Ordner übertragen wird. Es nimmt Ihnen lediglich die Arbeit ab, alle Dateien vorher auszuwählen.

Wenn Sie geänderte Dateien in Ihrer Arbeitskopie haben, die mittels `svn:externals` aus einem anderen Projektarchiv eingebunden wurden, können diese Änderungen nicht gleichzeitig übertragen

werden. Ein Warnsymbol erscheint unterhalb der Dateiliste, wenn das der Fall ist und ein Hinweistext erklärt, dass diese externen Dateien separat übertragen werden müssen.



Zu viele Dateien werden angezeigt

Falls Sie das Gefühl haben dass TortoiseSVN Ihnen viel zu viele Dateien im Übertragen-Dialog anzeigt die nicht unter Versionskontrolle stehen (z.B. vom Compiler erzeugte Dateien oder Sicherungsdateien vom Editor), haben Sie mehrere Möglichkeiten dies einzugrenzen. Sie können:

- diese Dateien im Eigenschaftsdialog in die Liste der auszuschließenden Dateien aufnehmen. Dies hat Einfluss auf *alle* Arbeitskopien gleichzeitig.
- Fügen Sie diese Dateien mit dem Befehl TortoiseSVN → Ignorieren zur Liste der ignorierten Dateien hinzu .Dadurch werden diese Dateien in die `svn:ignore` Eigenschaft des Ordners aufgenommen, in dem Sie den Befehl aufgerufen haben. Mit dem SVN Eigenschaften-Dialog können Sie die `svn:ignore` eines Verzeichnisses ändern.

Siehe [Abschnitt 4.13, „Ignorieren von Dateien und Ordnern“](#) für weitere Information.

Durch einen Doppelklick auf eine Datei im Übertragen-Dialog wird das Vergleichsprogramm gestartet, so dass Sie die Änderungen, welche Sie vorgenommen haben, genauer ansehen können. Das Kontextmenü bietet, wie Sie im Bild sehen können, weitere Optionen. Sie können von hier aus auch Dateien in eine andere Anwendung, z.B. einen Texteditor oder eine Entwicklungsumgebung ziehen.

Sie können Einträge an- oder abwählen, indem Sie das Kästchen links vom Eintrag markieren. Für Verzeichnisse können Sie **Umsch**-Auswahl verwenden, um die Auswahl rekursiv zu machen.

Die im unteren Bereich angezeigten Spalten können angepasst werden. Wenn Sie einen Rechtsklick auf einen Spaltenkopf machen, erscheint ein Kontextmenü aus dem Sie die anzuzeigenden Spalten auswählen können. Sie können auch die Spaltenbreiten anpassen, indem sie die Spaltenköpfe mit den Ziehmarken justieren. Diese Einstellungen werden gespeichert, so dass Sie beim nächsten Mal dieselben Spalten sehen.

Standardmäßig werden bei einer erfolgreichen Übertragung die von Ihnen gesperrten Dateien wieder freigegeben. Falls sie jedoch die Sperren behalten möchten, aktivieren Sie die Option **Sperren behalten**. Als Vorgabewert des Auswahlkästchen wird aus der Einstellung `no_unlock` in der Subversion Konfigurationsdatei verwendet. Lesen Sie in [Abschnitt 4.30.1, „Allgemeine Einstellungen“](#) nach, wie sie die Subversion Konfigurationsdatei bearbeiten können.



Ziehen und Ablegen

Sie können auch aus anderen Ordnern Dateien in den Übertragen-Dialog ziehen, solange die Arbeitskopie aus demselben Projektarchiv ausgecheckt wurde. Angenommen, Sie haben eine riesige Arbeitskopie und mehrere Explorerfenster mit Sichten auf verschiedene Ordner Ihrer Arbeitskopie geöffnet. Wenn Sie die Übertragung nicht aus dem obersten Ordner heraus starten wollen, weil das Durchsuchen der Arbeitskopie nach geänderten Dateien seine Zeit dauert, können Sie die Übertragung in einem Ordner starten und weitere Objekte aus den anderen Ordnern in den Dialog ziehen, welche dann auf einen Schlag mit übertragen werden.

Sie können unversionierte Dateien, die sich in einer Arbeitskopie befinden, direkt in den Übertragen-Dialog ziehen. Diese Dateien werden dann automatisch zur Versionskontrolle hinzugefügt.



Externes Umbenennen reparieren

Manchmal werden Dateien außerhalb von Subversion umbenannt, und sie werden in der Dateiliste als eine fehlende und eine nicht-versionierte Datei angezeigt. Damit sie die Historie der Datei nicht verlieren, müssen Sie Subversion über die Umbenennung informieren. Markieren Sie einfach beide, die alte (fehlende) und die neue (unversionierte) Datei und wählen Sie Kontextmenü → Umbenennen reparieren, um die beiden Dateien zu einer Umbenennung zusammenzufassen.

4.4.2. Änderungslisten

Der Übertragen-Dialog unterstützt die Änderungslisten von Subversion, mit der zueinander gehörende Dateien gruppiert werden können. Eine Beschreibung dieser Funktion findet sich in [Abschnitt 4.8](#), „Änderungslisten“.

4.4.3. Objekte vom Übertragen ausschließen

Manchmal arbeiten Sie mit versionierten Dateien, die sich häufig ändern, die Sie aber nicht übertragen wollen. Eventuell deutet das auf einen Mangel in Ihrem Erstellungsprozess hin - Warum sind diese Dateien versioniert? sollten Sie nicht besser mit Vorlagen arbeiten? Aber manchmal ist dies unvermeidlich. Ein klassischer Grund ist, dass Ihre Entwicklungsumgebung bei jedem Generieren einen Zeitstempel in der Projektdatei verändert. Die Projektdatei muss versioniert sein, da sie all zum Generieren erforderlichen Einstellungen enthält, jedoch muss sie nicht übertragen werden, nur weil sich der Zeitstempel geändert hat.

Um in diesen heiklen Fällen zu helfen, gibt es eine reservierte Änderungsliste namens `ignore-on-commit`. Jede zu dieser Änderungsliste hinzugefügte Datei, wird im Übertragen-Dialog nicht markiert. Sie können die Änderungen übertragen, müssen Die Datei aber von Hand wählen.

4.4.4. Logmeldungen

Geben Sie eine Logmeldung ein, die die Änderungen, die Sie übertragen genau beschreibt. Das vereinfacht es Ihnen später erheblich, Ihre Änderungen nachzuvollziehen, wenn Sie das Projekt durchsuchen. Die Logmeldung kann so lang oder so knapp sein wie sie möchten. Manche Projekte haben sogar genaue Richtlinien, in welcher Sprache und in welchem Format Logmeldungen verfasst werden müssen.

Sie können Ihre Logmeldungen mit einfachen Formatierungen versehen. Dazu wird eine Konvention ähnlich derer in e-mails angewendet. Um `text` zu formatieren, verwenden Sie `*text*` für fett, `_text_` für unterstrichen, und `^text^` für kursiv.

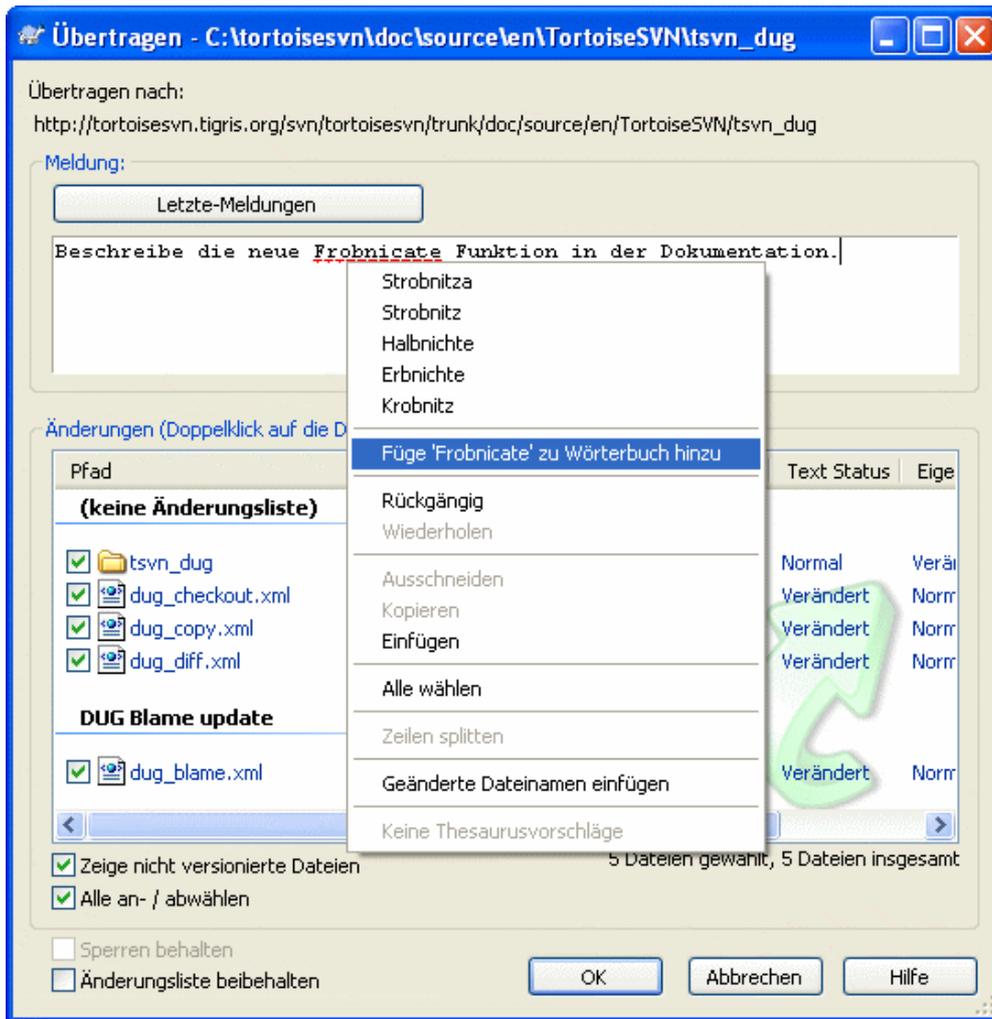


Abbildung 4.9. Rechtschreibprüfung beim Eingeben einer Logmeldung

TortoiseSVN enthält eine Rechtschreibprüfung die Sie bei der Korrektur Ihrer Logmeldungen unterstützt. Sie hebt sämtliche falsch geschriebenen Wörter hervor. Verwenden Sie das Kontextmenü, um auf die Korrekturvorschläge zuzugreifen. Verständlicherweise kennt die Rechtschreibprüfung nicht *jeden* technischen Ausdruck, den Sie verwenden, so dass manchmal korrekt geschriebene Wörter als fehlerhaft markiert werden. Aber keine Sorge. Sie können Diese Wörter über das Kontextmenü zu Ihrem persönlichen Wörterbuch hinzufügen.

Das Eingabefeld verfügt außerdem über eine automatisch Vervollständigende Funktion für Datei- und Funktionsnamen. Diese verwendet reguläre Ausdrücke, um Klassen- und Funktionsnamen aus den zu übertragenden Dateien sowie den Dateinamen selbst zu extrahieren. Sobald Sie die ersten drei Zeichen eines Wortes (oder **Strg+Leer**) eingeben haben, wird, falls es Übereinstimmungen gibt, eine Liste angezeigt, aus der Sie das vollständige Wort auswählen können. Die zusammen mit TortoiseSVN installierten regulären Ausdrücke finden sich im bin Ordner. Sie können obendrein Ihre eigenen regulären Ausdrücke definieren und in der Datei %APPDATA%\TortoiseSVN\autolist.txt abspeichern. Diese private Liste wird beim Aktualisieren von TortoiseSVN nicht überschrieben. Wenn Sie sich mit regulären Ausdrücken nicht auskennen, finden Sie ausführliche Informationen unter http://de.wikipedia.org/wiki/Regulärer_Ausdruck [http://de.wikipedia.org/wiki/Regul%C3%A4rer_Ausdruck/] sowie eine Online-Dokumentation und -Anleitung unter <http://www.regular-expressions.info/>.

Sie können die zuletzt verwendeten Logmeldungen wiederverwenden. Dazu klicken Sie auf **Letzte Meldungen** und wählen einen Eintrag aus der Liste der Meldungen für dieses Projektarchiv aus. Die Anzahl der zu speichernden Logmeldungen kann in den Einstellungen festgelegt werden.

Auf der **Gespeicherte Daten** Seite in den TortoiseSVN Einstellungen können Sie alle gespeicherten Logmeldungen auf einmal löschen. Alternativ können Sie im **Letzte Meldungen** Fenster Meldungen markieren und mittels **Entfernen** einzeln löschen .

Über das Kontextmenü können Sie per Namensliste einfügen die Liste der markierten Pfade in die Logmeldung einfügen.

Eine weitere Möglichkeit besteht darin, die Dateien einfach aus der Dateiliste in das Eingabefeld zu ziehen.



Spezielle Ordneigenschaften

Es gibt mehrere spezielle Ordneigenschaften welche z.B. für die Formatierung von Logmeldungen und die Rechtschreibkorrektur verwendet werden können. Sehen Sie dazu [Abschnitt 4.17, „Projekt-Einstellungen“](#).



Integration mit Fehlerverfolgungssystem

Wenn Sie Fehlerverfolgung aktiviert haben, können Sie in das Fehler-ID / Eintrags-Nr: Feld eine oder mehrere Eintragsnummer(n) Ihres Fehlerverfolgungssystems eingeben. Mehrere Einträge müssen durch Kommata getrennt werden. Alternativ können Sie, bei Verwendung von regulären Ausdrücken, die Eintragsnummern direkt in die Logmeldung einfließen lassen. Mehr dazu finden Sie in [Abschnitt 4.28, „Integration mit einem System zur Fehlerverfolgung“](#).

4.4.5. Fortschrittsdialog

Wenn Sie schlussendlich auf OK klicken wird die Übertragung gestartet und der Fortschrittsdialog angezeigt.

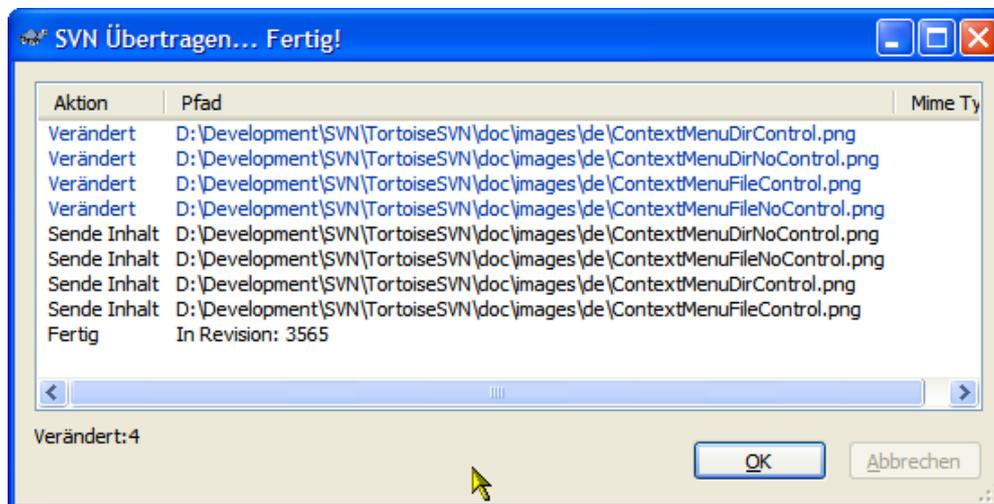


Abbildung 4.10. Eine laufende Übertragung im Fortschritts-Dialog

Der Fortschrittsdialog verwendet eine Farbcodierung, um verschiedene Aktionen anzuzeigen

Blau

Übertrage eine Änderung

Purpur

Übertragen eines neuen Objekts.

Dunkelrot

Übertragen einer Löschung oder Ersetzen eines Objekts.

Schwarz

Alle anderen Objekte.

Dies ist das Standard Farbschema, aber Sie können die Farben im Einstellungsdialog anpassen. Siehe [Abschnitt 4.30.1.4, „TortoiseSVN Farben“](#) für weitere Informationen.

4.5. Aktualisieren der Arbeitskopie mit Änderungen von anderen

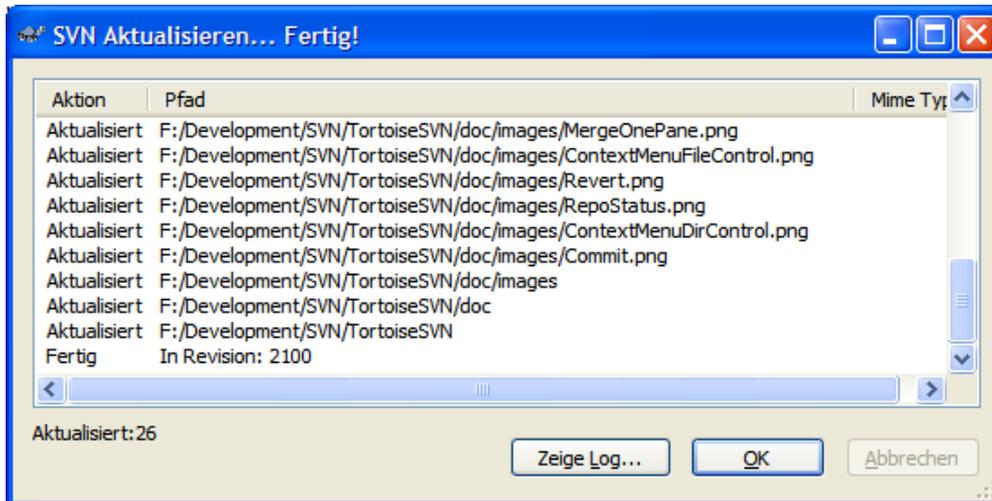


Abbildung 4.11. Der Fortschritts-Dialog nach Abschluss der Aktualisierung

Änderungen von anderen sollten Sie regelmäßig in Ihre eigene Arbeitskopie einfügen. Änderungen von anderen in die eigene Arbeitskopie einfügen wird auch *Aktualisieren* genannt. Sie können einzelne Dateien, mehrere Dateien, ganze Ordner oder gleich die komplette Arbeitskopie aktualisieren. Wählen Sie einfach die Dateien oder Ordner aus, welche Sie aktualisieren möchten und öffnen Sie dann das Kontextmenü durch einen Rechtsklick. Im Kontextmenü wählen Sie dann den Befehl TortoiseSVN → Aktualisieren aus und ein Fortschrittsdialog wird erscheinen. Änderungen von Anderen werden in Ihre Arbeitskopie eingefügt, wobei Ihre eigenen Änderungen selbstverständlich beibehalten werden. Das Projektarchiv selbst wird durch eine Aktualisierung nicht verändert.

Der Fortschrittsdialog verwendet eine Farbcodierung, um verschiedene Aktionen anzuzeigen

Purpur

Neues Objekt zur Arbeitskopie hinzugefügt.

Dunkelrot

Überflüssiges Objekt das aus der Arbeitskopie gelöscht wurde oder fehlendes Objekt, das in der Arbeitskopie ersetzt wurde.

Grün

Änderungen aus dem Projektarchiv wurden erfolgreich mit Ihren lokalen Änderungen zusammengeführt.

Hellrot

Änderungen aus dem Projektarchiv wurden mit Ihren lokalen Änderungen zusammengeführt. Es gab jedoch Konflikte, die Sie noch auflösen müssen.

Schwarz

Unverändertes Objekt in Ihrer Arbeitskopie, das durch ein neueres Objekt aus dem Projektarchiv ersetzt wurde

Dies ist das Standard Farbschema, aber Sie können die Farben im Einstellungsdialog anpassen. Siehe [Abschnitt 4.30.1.4, „TortoiseSVN Farben“](#) für weitere Informationen.

Falls Sie *Konflikte* während einer Aktualisierung erhalten (Dies kann passieren wenn jemand anders dieselbe Stelle in einer Datei geändert hat wie Sie und diese Änderungen nicht zusammenpassen) dann zeigt der Fortschritts-Dialog diese Dateien rot markiert an. Ein Doppelklick auf diese roten Einträge öffnet einen Konflikteditor, mit dem Sie solche Konflikte ganz einfach von Hand auflösen können.

Sobald die Aktualisierung beendet ist, zeigt der Fortschrittsdialog eine Zusammenfassung der aktualisierten, hinzugefügten und gelöschten Objekte unterhalb der Dateiliste an. Diese Information kann mittels **Strg+C** in die Zwischenablage kopiert werden.

Der normale „Aktualisieren“ Befehl hat keine Parameter und aktualisiert Ihre Arbeitskopie stets zur HEAD Revision des Projektarchivs, was der häufigste Anwendungsfall ist. Wenn Sie genau festlegen wollen, in welcher Revision sich Ihre Arbeitskopie befinden soll, sollten Sie den TortoiseSVN → Aktualisieren zu Revision... Befehl verwenden, der Ihnen genau das ermöglicht. Nehmen wir an, Ihre Arbeitskopie befindet sich in Revision 100, aber sie möchten sie gerne in Revision 50 zurückversetzen - Aktualisieren sie einfach zu Revision 50. In diesem Dialog können Sie auch die *Tiefe* beim Aktualisieren festlegen. Die verschiedenen Tiefen werden in [Abschnitt 4.3.1, „Rekursionstiefe“](#) beschrieben. Die Vorgabe ist *Arbeitskopie* und erhält die vorhandene Einstellung. Weiterhin können Sie wählen, ob Sie externe Projekte (auf die mittels `svn:externals` verwiesen wird) einschließen oder ignorieren wollen.



Achtung

Wenn Sie eine Datei oder Ordner auf eine bestimmte Revision aktualisieren, sollten Sie keine Änderungen daran vornehmen. Sie werden „out of date“ Fehlermeldungen erhalten sobald Sie versuchen, diese Dateien oder Ordner zu übertragen! Wenn Sie Änderungen an Dateien rückgängig machen und mit einer vorherigen Version weiterarbeiten wollen, können Sie mit Hilfe des Log-Dialogs zu einer früheren Version zurückkehren. Lesen Sie in [Abschnitt B.4, „Revisionen im Projektarchiv rückgängig machen“](#) nach, welche Methoden Ihnen dazu zur Verfügung stehen.

Der Befehl TortoiseSVN → Aktualisieren zu Revision kann manchmal nützlich sein, um zu überprüfen, wie Ihr Projekt zu einem früheren Zeitpunkt aussah. Im Allgemeinen ist es jedoch keine gute Idee, einzelne Dateien in einen früheren Zustand zu versetzen, da dadurch Ihre Arbeitskopie inkonsistent wird. Wenn die Datei, die sie aktualisieren umbenannt wurde, kann es sogar passieren, dass sie aus Ihrer Arbeitskopie verschwindet, da zu einem früheren Zeitpunkt keine Datei dieses Namens im Projektarchiv existierte. Beachten Sie bitte, dass das überlagerte Symbol des Objekts den „normal“ Status anzeigt und das Objekt somit nicht von aktualisierten Objekten unterschieden werden kann.

Wenn Sie lediglich eine lokale Kopie einer alten Version einer Datei haben wollen, ist es besser, den Kontextmenü → Revision speichern unter... Befehl aus dem Log-Dialog dafür zu verwenden.



Mehrere Dateien/Ordner

Wenn Sie mehrere Dateien und Ordner im Explorer auswählen und dann Aktualisieren... ausführen, so werden alle Dateien und Ordner auf dieselbe Revision aktualisiert, sogar dann wenn zwischen den einzelnen Aktualisierungen jemand anders Änderungen zum Projektarchiv übertragen hat.



Lokale Datei existiert bereits

Manchmal wird die Aktualisierung mit einer Fehlermeldung abbrechen welche besagt dass eine Datei mit demselben Namen bereits existiert. Dies passiert wenn Subversion eine neu versionierte Datei auschecken möchte, aber eine Datei mit demselben Namen und Pfad bereits in Ihrer Arbeitskopie vorhanden ist. Subversion wird niemals eine solche Datei überschreiben - Sie könnten dadurch ja wichtige Daten verlieren.

Wenn Sie diese Fehlermeldung erhalten ist die Lösung sehr einfach: Benennen Sie die unversionierte Datei in Ihrer Arbeitskopie um. Danach führen Sie die Aktualisierung nochmals durch.

Wenn Sie diese Fehlermeldung immer wieder erhalten, benutzen Sie TortoiseSVN → Prüfe auf Änderungen, um eine Liste von allen problematischen Dateien zu erhalten. Dadurch können Sie alle möglichen Konflikte im Vorfeld erkennen.

4.6. Konflikte auflösen

Ab und an werden Sie einen *Konflikt* erhalten, wenn Sie Ihre Arbeitskopie aktualisieren oder zu einer anderen URL wechseln. Es gibt zwei Arten von Konflikten:

Dateikonflikte

Ein Dateikonflikt entsteht, wenn zwei (oder mehr) Entwickler dieselben Zeilen einer Datei geändert haben.

Baumkonflikte

Ein Baumkonflikt entsteht, wenn ein Entwickler eine Datei oder einen Ordner umbenannt, verschoben oder gelöscht hat, den ein anderer Entwickler ebenfalls umbenannt, verschoben, gelöscht oder bearbeitet hat.

4.6.1. Dateikonflikte

Ein Konflikt tritt dann auf wenn mehrere Personen die gleichen Zeilen in einer Datei verändert haben. Da Subversion nichts über Ihr Projekt weiß, überlässt es in solchen Fällen Ihnen, den Konflikt aufzulösen. Wann immer ein Konflikt auftritt, können Sie die entsprechende Datei öffnen und nach Zeilen suchen welche mit <<<<<<< beginnen. Die Zeile in Konflikt ist folgendermaßen markiert:

```
<<<<<<< Dateiname
    Ihre Änderungen
=====
    Code aus dem Projektarchiv
>>>>>>> Revision
```

Außerdem werden für jede Datei in Konflikt drei weitere Dateien erstellt:

filename.ext.mine

Dies ist die Datei, so wie Sie war bevor Sie Ihre Arbeitskopie aktualisierten. Das heißt es ist Ihre eigene Originaldatei, inklusive der Änderungen welche Sie selbst vorgenommen haben.

filename.ext.OLDREV

Dies ist die Datei wie Sie ursprünglich war, ohne jegliche Änderungen, auch ohne den Änderungen welche Sie selbst an der Datei vorgenommen haben.

filename.ext.rNEWREV

Dies ist die Datei, wie sie im Projektarchiv gerade aktuell ist, d.h. diese Datei hat die Änderungen von den anderen Mitarbeitern bereits integriert, jedoch noch nicht die Ihren.

Sie können nun entweder einen Konflikteditor benutzen, den Sie durch den Befehl TortoiseSVN → Konflikt bearbeiten aufrufen können oder Sie können die Datei mit einem normalen Texteditor ändern und den Konflikt auf diese Weise auflösen.

Anschließend müssen Sie Subversion noch mitteilen, dass Sie den Konflikt aufgelöst haben. Dies geschieht mit dem Befehl TortoiseSVN → Konflikt aufgelöst. Bitte beachten Sie dass dieser Befehl nicht den Konflikt selbst löst, sondern nur Subversion mitteilt dass Sie selbst den Konflikt bereits gelöst haben. Der Befehl macht nichts weiter als die drei zusätzlich erstellten Dateien `filename.ext.mine` und `filename.ext.r*` zu löschen, damit sie Ihre Änderungen in das Projektarchiv übertragen können.

Falls ein Konflikt zwischen Binärdaten besteht, versucht Subversion nicht, die Daten selbst zusammenzuführen. Die lokale Datei bleibt unverändert (exakt so, wie sie Ihrer letzten Änderung entspricht) und Sie erhalten `Dateiname.ext.r*` Dateien. Wenn Sie Ihre eigenen Änderungen verwerfen wollen, tun Sie das mit dem Rückgängig Befehl. Wenn Sie Ihre Version beibehalten und die Version im Projektarchiv überschreiben wollen, verwenden Sie den Konflikt aufgelöst Befehl und übertragen anschließend die Daten ins Projektarchiv

Sie können den Konflikt aufgelöst Befehl für mehrere Dateien verwenden, indem Sie den übergeordneten Ordner markieren und TortoiseSVN → Konflikt aufgelöst... aus dem Kontextmenü wählen. Dies öffnet einen Auswahldialog, in dem alle konfliktbehafteten Dateien aufgelistet sind. Wählen Sie die Dateien, die sie als aufgelöst markieren wollen.

4.6.2. Baumkonflikte

Ein Baumkonflikt entsteht, wenn ein Entwickler eine Datei oder einen Ordner umbenannt, verschoben oder gelöscht hat, den ein anderer Entwickler ebenfalls umbenannt, verschoben, gelöscht oder bearbeitet hat. Es gibt verschiedene Ursachen für Baumkonflikte und alle erfordern unterschiedliche Vorgehensweisen, um den Konflikt aufzulösen.

Wenn eine Datei in Subversion lokal gelöscht wird, wird sie auch aus dem lokalen Dateisystem gelöscht. Das bedeutet, dass kein überlagertes Symbol angezeigt werden kann, wenn sie Teil eines Baumkonfliktes ist und dass Sie den Konflikt nicht mit Hilfe eines Rechtsklicks auflösen können. Verwenden Sie stattdessen den Auf Änderungen prüfen Dialog, um den Konflikt bearbeiten zu können.

TortoiseSVN kann dabei helfen, Änderungen zusammenzuführen, aber es kann zusätzliche Arbeit erforderlich sein, um die Konflikte aufzulösen. Bedenken Sie, dass nach eine Aktualisierung die BASE der Arbeitskopie dem Inhalt des Projektarchivs entspricht. Wenn Sie eine Änderung nach dem Aktualisieren rückgängig machen, wird das Objekt in den Status des Projektarchivs zurückversetzt und nicht in den Zustand in dem sie begonnen haben, ihre eigenen Änderungen durchzuführen.

4.6.2.1. Lokal gelöscht, eingehende Änderung beim Aktualisieren

1. Entwickler A verändert die Datei `F00.c` und überträgt die Änderung ins Projektarchiv.
2. Entwickler B benennt gleichzeitig die Datei `F00.c` in seiner Arbeitskopie in `Bar.c` um oder löscht `F00.c` bzw. den Elternordner.

Eine Aktualisierung der Arbeitskopie von Entwickler B resultiert in einem Baumkonflikt:

- Die Datei `F00.c` wurde aus der Arbeitskopie gelöscht, ist aber gleichzeitig als *Baumkonflikt* markiert.
- Wenn ein Konflikt nicht von einem Löschen, sondern von einem Umbenennen herrührt, ist die Datei `Bar.c` als *hinzugefügt* markiert, enthält aber nicht die Änderungen von Entwickler A.

Entwickler B muss sich nun entscheiden, ob er die Änderungen von Entwickler A übernehmen möchte. Im Fall des Umbenennens kann er die Änderungen an `F00.c` in `Bar.c` zusammenführen. Für einfache Löschungen kann er das Objekt mit den Änderungen von Entwickler A beibehalten und das Löschen verwerfen. Oder er kann, indem er den Konflikt ohne weitere Aktionen als aufgelöst markiert, die Änderungen von Entwickler A verwerfen.

Der Dialog zum Auflösen von Konflikten bietet Ihnen an, die Änderungen zusammenzuführen, wenn er das Original der umbenannten Datei `Bar.c` finden kann. Abhängig davon, von wo aus die Aktualisierung angestoßen wurde, kann das nicht möglich sein.

4.6.2.2. Lokal geändert, eingehendes Löschen beim Aktualisieren

1. Entwickler A benennt die Datei `F00.c` in `Bar.c` um und überträgt die Änderung ins Projektarchiv.
2. Entwickler B verschiebt die Datei `F00.c` in seiner Arbeitskopie.

Oder im Fall eines verschobenen Ordners...

1. Entwickler A benennt den Elternordner `FooOrdner` in `BarFolder` um und überträgt die Änderung ins Projektarchiv.
2. Entwickler B verschiebt die Datei `Foo.c` in seiner Arbeitskopie.

Eine Aktualisierung der Arbeitskopie von Entwickler B führt zu einem Baumkonflikt. Für einen einfachen Dateikonflikt:

- Die Datei `Bar.c` wird als normale Datei zur Arbeitskopie hinzugefügt.
- Die Datei `Foo.c` ist als *hinzugefügt mit Historie* markiert und hat einen Baumkonflikt.

Für einen Ordnerkonflikt:

- `BarOrdner` wird als normaler Ordner zur Arbeitskopie hinzugefügt.
- Der Ordner `FooOrdner` ist als *hinzugefügt mit Historie* markiert und hat einen Baumkonflikt.

Die Datei `Foo.c` ist als *verändert* markiert.

Entwickler B muss nun entscheiden, ob er die Reorganisation durch Entwickler A übernehmen will und seine Änderungen in der entsprechenden Datei in der neuen Struktur zusammenführt oder ob er einfach die Änderungen von A rückgängig macht und die lokale Datei beibehält.

Um seine lokalen Änderungen mit der Umstrukturierung zusammenzuführen, muss Entwickler B zunächst herausfinden wie die konfliktbehaftete Datei `Foo.c` im Projektarchiv umbenannt wurde. Dies kann mit Hilfe des Log-Dialogs geschehen. Die Änderungen müssen manuell zusammengeführt werden, da es derzeit keine Möglichkeit gibt, dies zu automatisieren oder zu vereinfachen. Sobald die Änderungen übernommen wurden, ist der konfliktbehaftete Pfad überflüssig und kann gelöscht werden. Verwenden Sie dazu die Entfernen Schaltfläche im Konflikteditor. Damit wird dieser Konflikt als aufgelöst markiert.

Falls Entwickler B beschließt, dass die Änderungen von A falsch waren, muss er **Beibehalten** im Konflikteditor wählen. Dadurch wird der Konflikt als aufgeöst markiert, aber die Änderungen von Entwickler A müssen manuell zurückgenommen werden. Der Log-Dialog hilft Ihnen wieder dabei festzustellen, was verschoben wurde.

4.6.2.3. Lokal gelöscht, eingehende Löschung beim Aktualisieren

1. Entwickler A benennt die Datei `Foo.c` in `Bar.c` um und überträgt die Änderung ins Projektarchiv.
2. Entwickler B benennt `Foo.c` in `Bix.c` um.

Eine Aktualisierung der Arbeitskopie von Entwickler B resultiert in einem Baumkonflikt:

- Die Datei `Bix.c` ist als *hinzugefügt mit Historie* markiert.
- Die Datei `Bar.c` ist als *normal* markiert.
- Die Datei `Foo.c` ist als *gelöscht* markiert und hat einen Baumkonflikt.

Um diesen Konflikt aufzulösen, muss Entwickler B zunächst herausfinden wie die konfliktbehaftete Datei `Foo.c` im Projektarchiv umbenannt wurde. Dies kann mit Hilfe des Log-Dialogs geschehen.

Danach muss sich Entwickler B entscheiden, welchen neuen Dateinamen von `Foo.c` er übernehmen möchte, den eigenen oder den von Entwickler A vergebenen Namen.

Nachdem Entwickler B den Konflikt manuell aufgelöst hat, muss der Baumkonflikt mit der Schaltfläche im Konflikteditor als aufgelöst markiert werden.

4.6.2.4. Lokal fehlend, eingehende Änderung beim Aktualisieren

1. Entwickler A verändert im Stamm die Datei `Foo.c` um und überträgt die Änderung ins Projektarchiv.
2. Entwickler B, der auf einem Zweig arbeitet, benennt `Foo.c` in `Bar.c` um und überträgt die Änderung ins Projektarchiv.

Das Zusammenführen der Änderungen von Entwickler A im Stamm mit der Arbeitskopie von Entwickler B führt zu einem Baumkonflikt:

- Die Datei `Bar.c` befindet sich bereits mit dem Status *normal* in der Arbeitskopie.
- Die Datei `Foo.c` ist als *fehlend* markiert und hat einen Baumkonflikt.

Um diesen Konflikt aufzulösen, muss Entwickler B den Dateikonflikt im Konflikteditor als aufgelöst markieren, wodurch er aus der Konfliktliste entfernt wird. Danach muss er entscheiden, ob er die fehlende Datei `Foo.c` aus dem Projektarchiv in die Arbeitskopie kopiert, die Änderungen von Entwickler A an `Foo.c` in die umbenannte Datei `Bar.c` überträgt oder die Änderungen ignoriert, indem er den Konflikt als aufgelöst markiert und nichts weiter unternimmt.

Beachten Sie, dass wenn Sie die fehlende Datei aus dem Projektarchiv kopieren und danach den Konflikt als aufgelöst markieren, ihre Kopie wieder entfernt wird. Sie müssen den Konflikt erst auflösen und danach die Datei kopieren.

4.6.2.5. Lokal bearbeitet, eingehende Löschung beim Zusammenführen

1. Entwickler A benennt im Stamm die Datei `Foo.c` in `Bar.c` um und überträgt die Änderung ins Projektarchiv.
2. Entwickler B, der auf einem Zweig arbeitet, modifiziert `Foo.c` und überträgt die Änderung ins Projektarchiv.

Es gibt einen äquivalenten Fall für verschobene Ordner, dieser wird aber in Subversion 1.6 noch nicht erkannt ...

1. Entwickler A benennt im Stamm den Ordner `FooOrdner` in `BarOrdner` um und überträgt die Änderung ins Projektarchiv.
2. Entwickler B, der auf einem Zweig arbeitet, verändert `Foo.c` in seiner Arbeitskopie.

Das Zusammenführen der Änderungen von Entwickler A im Stamm mit der Arbeitskopie von Entwickler B führt zu einem Baumkonflikt:

- Die Datei `Bar.c` ist als *hinzugefügt* markiert.
- Die Datei `Foo.c` ist als *verändert* markiert und hat einen Baumkonflikt.

Entwickler B muss nun entscheiden, ob er die Reorganisation durch Entwickler A übernehmen will und seine Änderungen in der entsprechenden Datei in der neuen Struktur zusammenführt oder ob er einfach die Änderungen von A rückgängig macht und die lokale Datei beibehält.

Um die lokalen Änderungen mit der Reorganisation zusammenzuführen, muss Entwickler B zunächst herausfinden wie die konfliktbehaftete Datei `Foo.c` im Projektarchiv umbenannt wurde. Dies kann mit Hilfe des Log-Dialogs für die Quelle des Zusammenführens geschehen. Der Konflikteditor zeigt nur das Log für die Arbeitskopie an, da er nicht wissen kann, welcher Pfad zum Zusammenführen genutzt wurde. Aus diesem Grund müssen Sie das selbst herausfinden. Die Änderungen müssen manuell zusammengeführt werden, da es derzeit keine Möglichkeit gibt, dies zu automatisieren oder zu vereinfachen. Sobald die Änderungen übernommen wurden, ist der konfliktbehaftete Pfad überflüssig und kann gelöscht werden. Verwenden Sie dazu die **Entfernen** Schaltfläche im Konflikteditor. Damit wird dieser Konflikt als aufgelöst markiert.

Falls Entwickler B beschließt, dass die Änderungen von A falsch waren, muss er **Beibehalten** im Konflikteditor wählen. Dadurch wird der Konflikt als aufgelöst markiert, aber die Änderungen von Entwickler A müssen manuell zurückgenommen werden. Der Log-Dialog hilft Ihnen wieder dabei festzustellen, was verschoben wurde.

4.6.2.6. Lokal gelöscht, eingehende Löschung beim Zusammenführen

1. Entwickler A benennt im Stamm die Datei `Foo.c` in `Bar.c` um und überträgt die Änderung ins Projektarchiv.

- Entwickler B, der auf einem Zweig arbeitet, benennt `Foo.c` in `Bix.c` um und überträgt die Änderung ins Projektarchiv.

Das Zusammenführen der Änderungen von Entwickler A im Stamm mit der Arbeitskopie von Entwickler B führt zu einem Baumkonflikt:

- Die Datei `Bix.c` ist als *normal* (unverändert) markiert.
- Die Datei `Bar.c` ist als *hinzugefügt mit Historie* markiert.
- Die Datei `Foo.c` ist als *fehlend* markiert und hat einen Baumkonflikt.

Um diesen Konflikt aufzulösen, muss Entwickler B zunächst herausfinden wie die konfliktbehaftete Datei `Foo.c` im Projektarchiv umbenannt wurde. Dies kann mit Hilfe des Log-Dialogs für die Quelle des Zusammenführens geschehen. Der Konflikteditor zeigt nur das Log für die Arbeitskopie an, da er nicht wissen kann, welcher Pfad zum Zusammenführen genutzt wurde. Aus diesem Grund müssen Sie das selbst herausfinden.

Danach muss sich Entwickler B entscheiden, welchen neuen Dateinamen von `Foo.c` er übernehmen möchte, den eigenen oder den von Entwickler A vergebenen Namen.

Nachdem Entwickler B den Konflikt manuell aufgelöst hat, muss der Baumkonflikt mit der Schaltfläche im Konflikteditor als aufgelöst markiert werden.

4.7. Statusinformationen anzeigen

Während Sie an Ihrem Projekt arbeiten müssen Sie oft wissen, welche Dateien geändert wurden, welche Sie neu hinzugefügt haben oder welche Sie gelöscht haben, oder auch welche Dateien von anderen geändert und zum Projektarchiv übertragen wurden.

4.7.1. Überlagerte Symbole

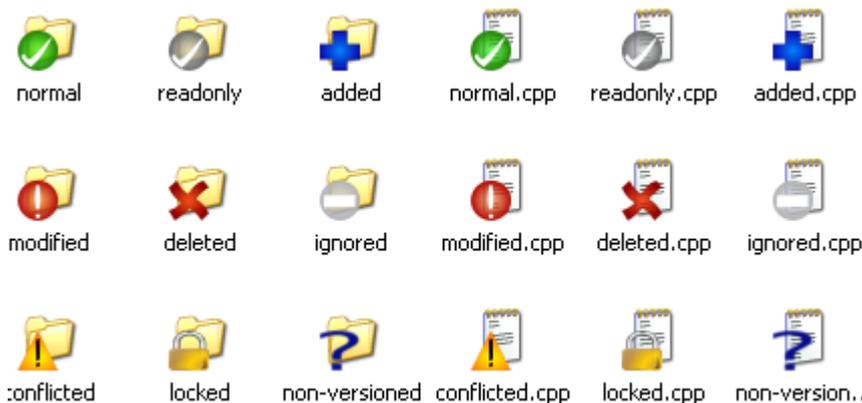


Abbildung 4.12. Explorer mit überlagerten Symbolen

Nun da Sie eine frisch ausgecheckte Arbeitskopie eines Subversion Projektarchivs haben zeigt Ihnen der Explorer diese Dateien und Ordner mit leicht geänderten Symbolen an. Dies ist mit ein Grund weshalb TortoiseSVN so populär ist. TortoiseSVN fügt jedem Datei- / Ordnersymbol ein kleines überlagertes Symbol hinzu. Abhängig vom Subversion Status wird ein unterschiedliches Symbol überlagert angezeigt.



Eine frisch ausgecheckte Arbeitskopie hat nur überlagerte Symbole mit grünem Haken. Dies bedeutet dass der Subversion Status *normal* ist.



Sobald Sie eine Datei ändern, ändert sich auch der Status der Datei auf *verändert* und das überlagerte Symbol ändert sich in ein rotes Ausrufezeichen. Auf diese Weise können Sie mit einem Blick feststellen, welche Dateien Sie geändert und noch nicht in das Projektarchiv übertragen haben.



Falls während einer Aktualisierung ein *Konflikt* auftrat, so werden solche Dateien mit einem gelben Ausrufezeichen markiert.



Wenn eine Datei die `svn:needs-lock` Eigenschaft besitzt, setzt Subversion den Schreibschutz für diese Datei, bis Sie eine Sperre für die Datei holen. Schreibgeschützte Dateien erhalten dieses Symbol, um anzuzeigen, dass Sie die Datei erst sperren müssen, bevor Sie sie bearbeiten können.



Wenn Sie die Sperre für eine Datei besitzen und der Subversion Status *normal* ist, erinnert Sie dieses Symbol daran, dass Sie die Sperre wieder freigeben müssen, wenn Sie sie nicht benötigen, damit andere ihre Änderungen übertragen können.



Dieses überlagerte Symbol zeigt, dass Dateien oder Ordner zum *Löschen* aus der Versionskontrolle markiert wurden oder dass TortoiseSVN eine Datei unter Versionskontrolle vermisst.



Das Pluszeichen bedeutet, dass eine Datei oder ein Ordner neu zur Versionskontrolle *hinzugefügt* wurde.



Das Balkensymbol bedeutet, dass eine Datei oder ein Ordner von der Versionskontrolle *ignoriert* wird. Dieses Symbol ist optional.



Dieses Symbol wird für Dateien und Ordner verwendet, die sich weder unter Versionskontrolle befinden noch ignoriert sind. Das Symbol ist optional.

Sie werden möglicherweise feststellen, dass nicht alle diese Symbole auf Ihrem Rechner dargestellt werden. Das liegt daran, dass Windows die Anzahl der überlagerten Symbole beschränkt. Windows selber verwendet einige, und wenn Sie gleichzeitig eine ältere Version von TortoiseCVS installiert haben, sind nicht genügend Plätze für die Symbole beider Anwendungen frei. Deshalb versucht TortoiseSVN ein „Guter Bürger(tm)“ zu sein und schränkt seine Verwendung von überlagerten Symbolen ein, damit andere Anwendungen eine Chance haben.

Da es mittlerweile viele Tortoise Clients gibt (TortoiseCVS, TortoiseHG, ...) wird die Beschränkung der überlagerten Symbole zu einem echten Problem. Als Lösungsansatz wurde im TortoiseSVN Projekt ein gemeinsamer Satz von Symbolen, der als DLL geladen wird und von allen Tortoise Clients genutzt werden kann, implementiert. Fragen Sie Ihren Tortoise Anbieter, ob er die gemeinsam genutzten Symbole bereits unterstützt.

Eine Beschreibung, wie die überlagerten Symbole mit dem Subversion Status zusammenhängen und weitere technische Details finden sich in [Abschnitt F.1, „Überlagerte Symbole“](#).

4.7.2. TortoiseSVN Spalten im Windows Explorer

Die Informationen, die durch die überlagerten Symbole angezeigt werden, können neben weiteren Informationen auch als zusätzliche Spalten in der Explorer Detailansicht dargestellt werden.

Klicken Sie einfach eine der vorhandenen Spalten mit der rechten Maustaste an und wählen Sie Weitere... aus dem Kontextmenü. In diesem Dialog können Sie die Spalten festlegen, die in der Ansicht „Details“ angezeigt werden sollen. Die Einträge, die mit SVN beginnen gehören zu TortoiseSVN. Setzen Sie den Haken vor jeden Eintrag, den Sie wünschen. Klicken Sie dann auf OK. Die Spalten werden rechts an die bestehenden angehängt. Nutzen Sie Ziehen und Ablegen, um Sie an die gewünschte Position zu bekommen.



Wichtig

Die zusätzlichen Spalten im Windows Explorer stehen unter Vista nicht mehr zur Verfügung, da Microsoft beschlossen hat, solche Spalten nicht mehr für *alle* sondern nur für bestimmte Dateitypen zuzulassen.



Tipp

Wenn das aktuelle Layout in allen Arbeitskopien angezeigt werden soll, müssen Sie es als Standard-Layout festlegen.

4.7.3. Prüfe auf Änderungen

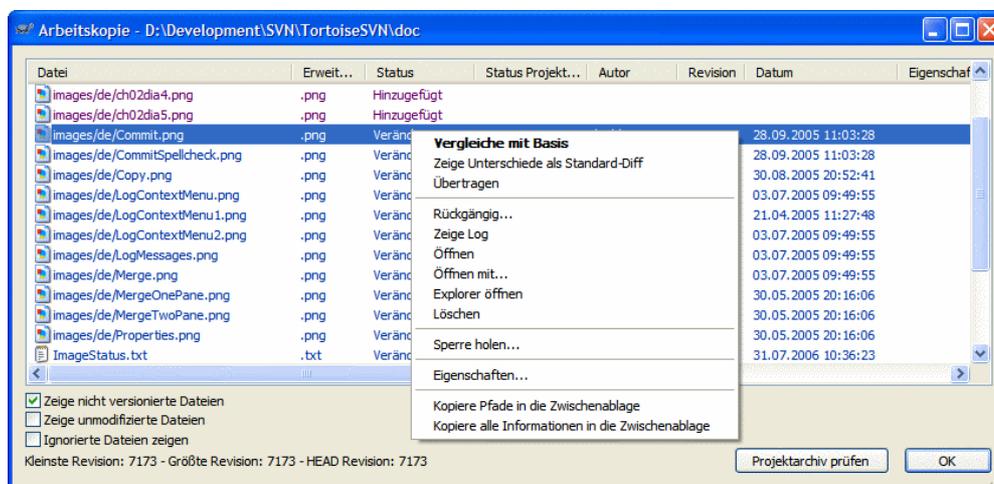


Abbildung 4.13. Prüfe auf Änderungen

Oft ist es nützlich zu wissen, welche Dateien von anderen bereits geändert und im Projektarchiv gespeichert wurden. Dazu gibt es den Befehl TortoiseSVN → Auf Änderungen überprüfen.... Dieser Dialog zeigt Ihnen alle veränderten oder unversionierten Dateien in Ihrer Arbeitskopie.

Mit der Projektarchiv prüfen Schaltfläche können Sie das Projektarchiv auf Änderungen prüfen. Auf diese Weise können Sie vor dem Aktualisieren feststellen, ob es möglicherweise Konflikte gibt. Sie können aus diesem Dialog heraus auch einzelne Dateien - ohne den gesamten Ordner - aus dem Projektarchiv aktualisieren. Standardmäßig holt Projektarchiv prüfen nur den Status des Projektarchivs entsprechend der Tiefe der Arbeitskopie. Wenn Sie alle Dateien und Ordner des Projektarchivs, inklusive derer, die Sie nicht ausgecheckt haben, sehen möchten, halten Sie die **Umsch** Taste gedrückt, während Sie auf Projektarchiv prüfen klicken.

Der Dialog verwendet eine Farbcodierung, um verschiedene Status anzuzeigen.

Blau

Lokal veränderte Objekte.

Purpur

Hinzugefügte Objekte. Objekte, die mit Historie hinzugefügt wurden, erhalten ein + Zeichen in der Spalte **Text Status** und ein Hinweistext zeigt an woher die Kopie stammt.

Dunkelrot

Gelöschte oder fehlende Objekte

Grün

Objekte, die lokal und im Projektarchiv verändert wurden. Die Änderungen werden beim Aktualisieren zusammengeführt. Diese Objekte *können* Konflikte beim Aktualisieren erzeugen.

Hellrot

Lokal veränderte Objekte, die im Projektarchiv gelöscht wurden oder im Projektarchiv geänderte Objekte, die lokal gelöscht wurden. Diese Objekte *werden* Konflikte beim Aktualisieren erzeugen.

Schwarz

Unveränderte und nicht versionierte Objekte

Dies ist das Standard Farbschema, aber Sie können die Farben im Einstellungsdialog anpassen. Siehe [Abschnitt 4.30.1.4, „TortoiseSVN Farben“](#) für weitere Informationen.

Objekte, die zu einem anderen Pfad im Projektarchiv gewechselt wurden, werden durch ein (s) gekennzeichnet. Vielleicht haben Sie während der Entwicklung etwas auf einen Zweig umgeschaltet und vergessen, zurück zu trunk zu wechseln. Dies ist Ihr Warnzeichen!

Aus dem Kontextmenü des Dialoges heraus können Sie sich die Unterschiede anzeigen lassen. Wählen sie die lokalen Änderungen, die *Sie* gemacht haben, mittels **Kontextmenü → Vergleiche mit Basis**. Prüfen Sie die Änderungen im Projektarchiv, die andere gemacht haben, mittels **Kontextmenü Zeige Unterschiede als Standard-Diff**

Sie können auch Änderungen in einzelnen Dateien rückgängig machen. Falls Sie aus Versehen eine Datei gelöscht haben, wird sie in diesem Dialog als *Fehlend* angezeigt und kann mittels *Rückgängig* wieder hergestellt werden.

Nicht versionierte und ignorierte Dateien können von hier aus mittels **Kontextmenü → Löschendirekt** in den Papierkorb verschoben werden. Wenn Sie die Dateien unter Umgehung des Papierkorbes permanent löschen wollen, halten Sie die **Umsch** Taste gedrückt, während Sie **Löschen** klicken.

Wenn Sie sich eine Datei genauer anschauen wollen, können Sie diese von hier aus in einen Editor ziehen.

Die angezeigten Spalten können angepasst werden. Wenn Sie einen Rechtsklick auf einen Spaltenkopf machen, erscheint ein Kontextmenü aus dem Sie die anzuzeigenden Spalten auswählen können. Sie können auch die Spaltenbreiten anpassen, indem sie die Spaltenköpfe mit den Ziehmarken justieren. Diese Einstellungen werden gespeichert, so dass Sie beim nächsten Mal wieder dieselben Spalten sehen.

Wenn Sie gleichzeitig an mehreren voneinander unabhängigen Aufgaben arbeiten, können Sie Dateien in Änderungslisten zusammenfassen. Lesen Sie [Abschnitt 4.4.2, „Änderungslisten“](#) für weitere Informationen.

Am unteren Rand des Dialoges sehen Sie eine Zusammenfassung der Revisionen in Ihrer Arbeitskopie. Es handelt sich dabei um die *übertragenen* Revisionen, nicht die *aktualisierten*. Sie stellen den Revisionsbereich dar, in dem die Dateien zuletzt übertragen wurden. Beachten Sie bitte dass der angezeigte Bereich sich nur die dargestellten Objekte bezieht, nicht auf alle in der Arbeitskopie vorhandenen Objekte. Wenn sie diese Information sehen möchten, wählen Sie die Option **Zeige unmodifizierte Daten**.



Tipp

Wenn Sie eine flache Ansicht Ihrer Arbeitskopie (also alle Dateien und Ordner in einer Liste) wollen, dann ist der **Prüfe auf Änderungen**-Dialog der einfachste Weg das zu erreichen. Wählen Sie einfach die **Zeige unmodifizierte Dateien** Option, um alle Dateien in Ihrer Arbeitskopie anzuzeigen.



Externes Umbenennen reparieren

Manchmal werden Dateien außerhalb von Subversion umbenannt, und sie werden in der Dateiliste als eine fehlende und eine nicht-versionierte Datei angezeigt. Damit sie die Historie der Datei nicht verlieren, müssen Sie Subversion über die Umbenennung informieren. Markieren Sie einfach beide, die alte (fehlende) und die neue (unversionierte) Datei und wählen Sie **Kontextmenü → Umbenennen reparieren**, um die beiden Dateien zu einer Umbenennung zusammenzufassen.

4.7.4. Unterschiede anzeigen

Oft möchten Sie sehen, was Sie in einer Datei geändert haben. Das können Sie durch das Markieren einer Datei mit dem TortoiseSVN Kontextmenübefehl **Vergleiche** tun. Diese Funktion startet TortoiseMerge (oder ein externes Vergleichsprogramm), das Ihnen die lokalen Änderungen (gegenüber der BASE Revision) seit dem letzten Auschecken oder Aktualisieren anzeigt.



Tipp

Auch wenn Sie sich nicht in einer Arbeitskopie befinden oder mehrere Versionen einer Datei haben, können Sie diese vergleichen:

Wählen Sie die beiden Dateien, die Sie vergleichen wollen (z.B. mit **Strg und Maus**) und wählen Sie **Vergleiche unter TortoiseSVN im Kontextmenü**. Die zuletzt markierte Datei (die mit dem Fokus, der gestrichelte Rahmen) wird als die neuere betrachtet.

4.8. Änderungslisten

In einer idealen Welt würden Sie normalerweise nur an einer Aufgabe zur Zeit arbeiten und Ihre Arbeitskopie enthielte nur einen logischen Satz an Änderungen. Aber zurück zur Realität. Es kommt mit Sicherheit häufiger vor, dass Sie an mehreren unzusammenhängenden Aufgaben gleichzeitig arbeiten und beim Übertragen alle Änderungen im Dialog angezeigt bekommen. Die **Änderungslisten** ermöglichen es Ihnen, Dateien in Gruppen zusammenzufassen. Dies kann natürlich nur sinnvoll funktionieren, wenn sich die Änderungen nicht überschneiden. Wenn mehrere Änderungen dieselbe Datei betreffen, gibt es keine Möglichkeit, die Änderungen auseinander zu halten.



Wichtig

Die Änderungslisten stehen nur unter Windows XP und neuer zur Verfügung, da sie eine Shell-Funktionalität benötigen, die unter Windows 2000 nicht zur Verfügung steht. Sorry, Win2K ist mittlerweile recht alt, also bitte nicht beschweren.

Sie treffen Änderungslisten an verschiedenen Stellen an, die wichtigsten sind jedoch der „Übertragen“- und der **Prüfe auf Änderungen**-Dialog. Lassen Sie uns mit dem **Prüfe auf Änderungen**-Dialog beginnen, nachdem Sie an einigen Funktionen und vielen Dateien gearbeitet haben. Wenn Sie den Dialog

zum ersten Mal öffnen, werden alle geänderten Dateien zusammen angezeigt. Nehmen wir an, Sie möchten das Ganze organisieren und diese Dateien nach Funktionen gruppieren.

Wählen Sie eine oder mehrere Dateien und rufen Sie Kontextmenü → In Änderungsliste übernehmen auf, um ein Objekt zu einer Änderungsliste hinzuzufügen. Zu Anfang gibt es noch keine Änderungslisten. Deshalb wird beim ersten Mal eine neue Änderungsliste angelegt. Geben Sie ihr einen beschreibenden Namen und klicken Sie auf OK. Der Übertragen-Dialog wird nun Gruppen von Einträgen anzeigen.

Sobald Sie eine Änderungsliste angelegt haben, können Sie Objekte aus einer anderen Änderungsliste oder dem Windows Explorer auf diese ziehen und ablegen. Per ziehen und ablegen aus dem Explorer können Sie ein Objekt zur Änderungsliste hinzufügen, bevor es modifiziert wurde. Sie können das auch aus dem Prüfe auf Änderungen-Dialog heraus machen, müssen sich dann aber alle nicht modifizierten Dateien anzeigen lassen.

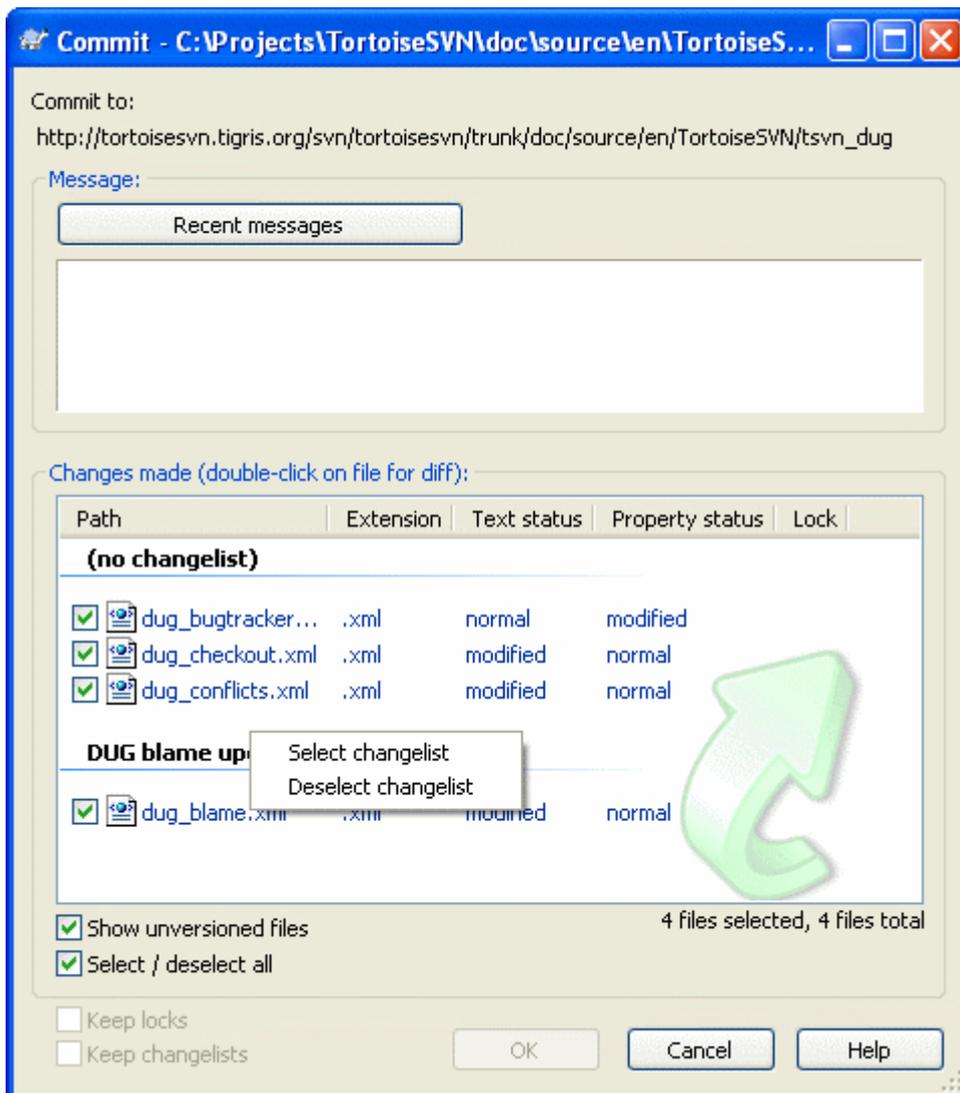


Abbildung 4.14. Der Übertragen-Dialog mit Änderungsliste

Im Übertragen-Dialog können Sie die gleichen Dateien nach Änderungsliste gruppiert sehen. Abgesehen davon, dass Sie einen optischen Hinweis auf Zusammenhänge erhalten, können Sie auch die Gruppenköpfe verwenden, um die zu übertragenden Dateien zu markieren.

Unter Windows XP erscheint, wenn Sie einen Rechtsklick auf einen Spaltenkopf machen, ein Kontextmenü mit dessen Hilfe Sie alle Gruppeneinträge (de)aktivieren können. Unter Vista ist dies jedoch

nicht nötig. Klicken Sie auf den Gruppenkopf, um alle Einträge zu wählen. Markieren Sie dann einen der gewählten Einträge um alle zu aktivieren.

TortoiseSVN reserviert eine Änderungsliste namens `ignore-on-commit` für den eigenen Gebrauch. Diese Änderungsliste wird zum Markieren von Dateien verwendet, die Sie meistens selbst dann nicht übertragen wollen, wenn sie lokale Änderungen enthalten. Diese Funktion wird in [Abschnitt 4.4.3](#), „Objekte vom Übertragen ausschließen“ beschrieben.

Wenn Sie Dateien übertragen, die zu einer Änderungsliste gehören, wird die Mitgliedschaft in der Änderungsliste normalerweise nicht mehr benötigt. Aus diesem Grund werden Dateien beim Übertragen automatisch aus der Änderungsliste entfernt. Wenn Sie das nicht wollen, aktivieren Sie die Option `Änderungsliste beibehalten` am Ende des Übertragen-Dialogs.



Tipp

Änderungslisten sind eine rein lokale Angelegenheit. Das Anlegen und Löschen von Änderungslisten wird weder das Projektarchiv noch eine andere Arbeitskopie beeinflussen. Sie sind lediglich ein bequemes Hilfsmittel zum Organisieren Ihrer Dateien.

4.9. Log-Dialog

Für jede Änderung, welche sie machen und zum Projektarchiv übertragen, müssen Sie eine Logmeldung eingeben, welche die Änderung beschreibt. Damit können Sie später herausfinden, welche Änderungen wer wann gemacht hat. Auf diese Weise haben Sie eine detaillierte Aufzeichnung über den Fortschritt Ihres Projektes.

Der Log-Dialog lädt alle diese Logmeldungen aus dem Projektarchiv und zeigt Ihnen diese, einschließlich einer Liste der Dateien welche in jeder Übertragung verändert wurden. Die Anzeige ist in drei Bereiche aufgeteilt.

- Der obere Bereich zeigt eine Liste von Revisionen in welchen Änderungen in der Datei/dem Ordner übertragen wurden. Diese Liste enthält Datum, Zeit und die Person, welche die Übertragung vorgenommen hat. Außerdem wird der erste Teil der Logmeldung angezeigt

Zeilen in blau zeigen an, dass etwas in dieser Revision kopiert/verschoben/umbenannt wurde (vielleicht von einem anderen Zweig).

- Der mittlere Bereich zeigt die ganze Logmeldung für die ausgewählte Revision an.
- Der untere Bereich zeigt eine Liste von Dateien und Ordnern welche in der ausgewählten Revision verändert wurden.

Aber das ist noch nicht alles - der Log-Dialog stellt auch viele Funktionen zur Verfügung mit denen Sie noch mehr und noch detailliertere Informationen über die Projektgeschichte herausfinden können.

4.9.1. Den Log-Dialog starten

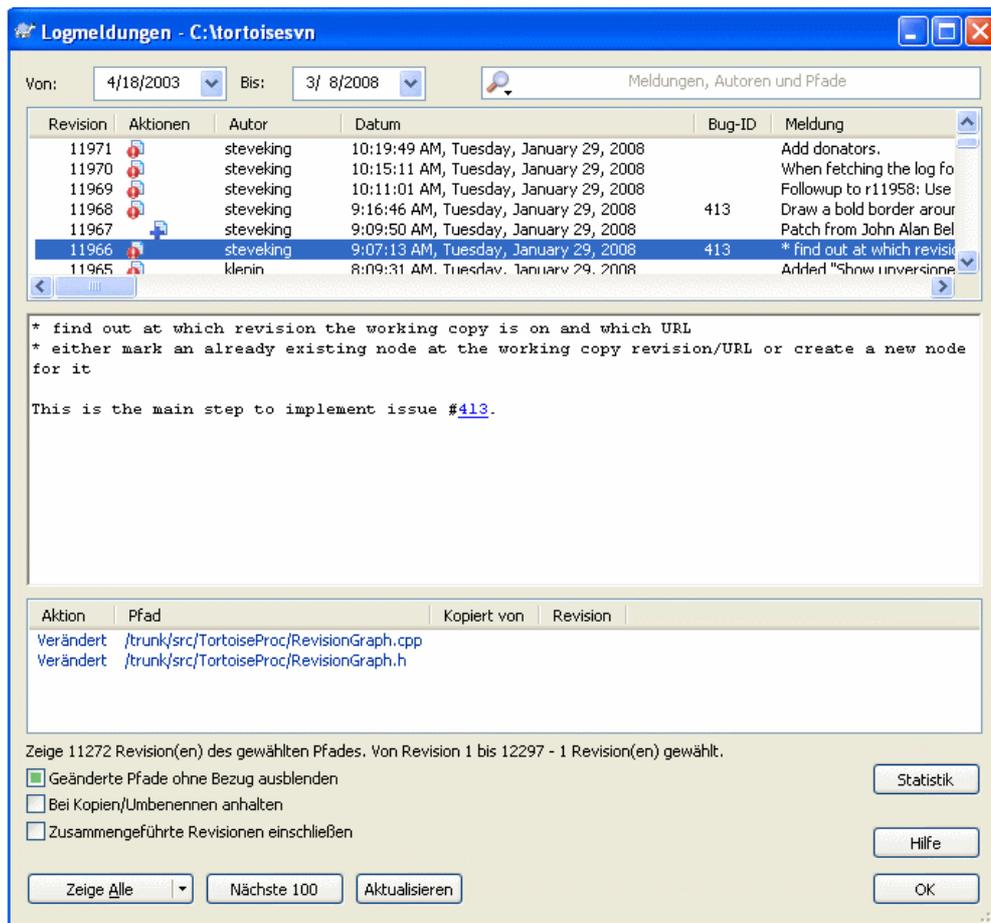


Abbildung 4.15. Der Log-Dialog

Es gibt mehrere Orte, von denen Sie den Log-Dialog starten können:

- Aus dem TortoiseSVN Kontextmenü.
- Aus der Eigenschaftsseite
- Aus dem Fortschritts-Dialog, nachdem eine Aktualisierung beendet wurde. Dann jedoch zeigt der Log-Dialog nur die Einträge, welche seit Ihrer letzten Aktualisierung gemacht wurden.

Falls das Projektarchiv nicht erreicht werden kann wird der **Offline gehen** Dialog, angezeigt, der in [Abschnitt 4.9.10](#), „Offline Modus“ beschrieben wird.

4.9.2. Aktionen im Revisionslog

Der obere Bereich enthält eine Aktionen Spalte, in der Symbole die Änderungen einer Revision zusammenfassen. Es gibt vier verschiedene Symbole, die jeweils in einer eigenen Spalte angezeigt werden.



in der ersten Spalte zeigt an, dass in einer Revision eine Datei oder ein Ordner *geändert* wurde.



in der zweiten Spalte zeigt an, dass in einer Revision eine Datei oder ein Ordner neu zur Versionskontrolle *hinzugefügt* wurde.



in der dritten Spalte zeigt an, dass in einer Revision eine Datei oder ein Ordner aus der Versionskontrolle *gelöscht* wurde.



in der vierten Spalte zeigt an, dass in einer Revision eine Datei oder ein Ordner *ersetzt* wurde.

4.9.3. Zusätzliche Informationen erhalten

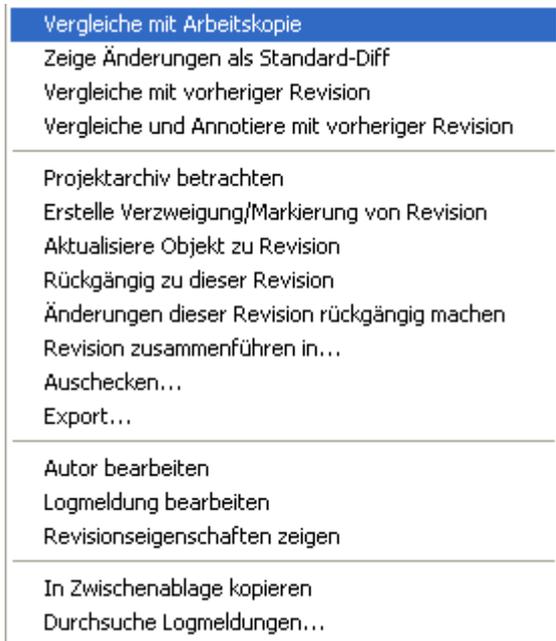


Abbildung 4.16. Das Kontextmenü des Log-Dialogs

Der obere Bereich des Log-Dialogs besitzt ein Kontextmenü das Ihnen Zugriff auf weitere Informationen ermöglicht. Einige Funktionen stehen nur bei Dateien, andere bei Ordnern zur Verfügung.

Vergleiche mit Arbeitskopie

Ein Vergleichsprogramm starten, um Ihnen die Änderungen, die seit der gewählten Revision vorgenommen worden, anzuzeigen. Das mit TortoiseSVN installierte Vergleichsprogramm ist TortoiseMerge. Wurde der Log-Dialog für ein Verzeichnis aufgerufen, so wird in TortoiseMerge eine Liste der geänderten Dateien angezeigt. Aus dieser Liste können Sie einen Eintrag wählen, um die Änderungen für jede Datei einzeln zu betrachten.

Vergleiche und Annotiere mit Arbeitskopie BASE

Nur für Dateien: Die gewählte Revision sowie die BASE Version annotieren und das Ergebnis in einem Vergleichsprogramm darstellen. Lesen Sie [Abschnitt 4.23.2, „Unterschiede annotieren“](#) für weitere Informationen.

Unterschiede als Standard-Diff anzeigen

Die Änderungen an allen Dateien in der gewählten Revision im Standard-Diff (GNU Patch) Format anzeigen. Dieses Format zeigt nur die Unterschiede mit ein paar Zeilen Kontext an. Es ist schwieriger zu lesen als ein visueller Vergleich, aber es zeigt alle Änderungen auf einmal.

Vergleiche mit der vorherigen Revision

Die gewählte Revision mit der vorherigen Revision vergleichen. Dies funktioniert ähnlich wie ein Vergleich mit Ihrer Arbeitskopie. Bei Ordnern werden zunächst die geänderten Dateien angezeigt, so dass Sie die Dateien für den Vergleich auswählen können.

Vergleiche und Annotiere mit vorheriger Revision

Die Liste der geänderten Dateien anzeigen, um die zu vergleichenden Dateien auszuwählen. Die gewählte sowie die vorherige Revision annotieren und das Ergebnis in einem Vergleichsprogramm darstellen. (Nur für Ordner)

Speichere Revision unter...

Nur für Dateien: Die gewählte Revision speichern, so dass Sie eine ältere Version der Datei erhalten.

Öffnen / Öffnen mit...

Nur für Dateien: Die markierte Datei entweder mit dem Standardprogramm für den Dateityp oder mit einem von Ihnen gewählten Programm öffnen.

Annotieren...

Nur für Dateien: Die Datei bis zur gewählten Revision annotieren.

Das Projektarchiv betrachten

Den Projektarchivbetrachter öffnen, um das markierte Objekt in der gewählten Revision des Projektarchivs zu betrachten.

Erstelle Zweig/Marke von Revision

Einen Zweig oder eine Marke für die gewählte Revision erstellen. Dies ist nützlich für den Fall, dass sie vergessen haben eine bestimmte Version Ihres Projektes zu markieren und Sie seitdem bereits weitere Änderungen in das Projektarchiv übertragen haben.

Aktualisiere zu Revision

Ihre Arbeitskopie zu einer bestimmten Revision aktualisieren. Nützlich wenn Sie möchten, dass Ihre Arbeitskopie einem Zustand in der Vergangenheit entspricht oder wenn es weitere Übertragungen im Projektarchiv gibt, die Sie schrittweise übernehmen wollen. Es ist am besten, ein komplettes Arbeitsverzeichnis und nicht nur eine einzelne Datei zu aktualisieren, da andernfalls Ihre Arbeitskopie inkonsistent sein wird und Sie keine Änderungen übertragen können.

Falls Sie eine frühere Änderung vollständig zurücknehmen wollen, verwenden Sie stattdessen *Rückgängig zu dieser Revision*.

Rückgängig zu dieser Revision

Zu einer früheren Revision zurückkehren. Wenn Sie mehrere Änderungen vorgenommen haben und nun beschließen, dass Sie den Zustand in Revision N wieder herstellen wollen, ist das der Befehl Ihrer Wahl. Wieder werden die Änderungen nur in Ihrer Arbeitskopie rückgängig gemacht. Die Aktion beeinflusst das Projektarchiv zunächst *nicht*, bis Sie die Änderungen wieder übertragen haben. Beachten Sie, dass diese Aktion *alle* Änderungen nach der gewählten Revision rückgängig macht und das Objekt durch die gewählte Revision ersetzt.

Falls Ihre Arbeitskopie vor dieser Aktion unmodifiziert war, wird sie anschließend das modifizierte Symbol anzeigen. Falls Sie bereits lokale Änderungen vorgenommen haben, wird dieser Befehl die *Rückgängig*-Änderungen in Ihrer Arbeitskopie zusammenführen.

Intern führt Subversion alle Änderungen seit dieser Revision rückwärts zusammen, so dass sie insgesamt zurückgenommen werden.

Falls Sie beschließen, dass Sie die *Rückgängig Aktion rückgängig machen* und Ihre Arbeitskopie in ihren vorherigen Zustand versetzen wollen, sollten Sie **TortoiseSVN → Rückgängig** aus dem Explorer-Kontextmenü wählen. Dadurch werden die durch das rückwärts Zusammenführen erzeugten lokalen Änderungen verworfen.

Wenn Sie nur sehen möchten, wie eine Datei oder ein Ordner in einer früheren Revision aussahen, rufen Sie stattdessen *Aktualisiere zu Revision* oder *Speichere Revision unter...* auf.

Änderungen dieser Revision rückgängig machen

Änderungen in einer bestimmten Revision rückgängig machen. Die Änderungen werden in Ihrer Arbeitskopie rückgängig gemacht, das Projektarchiv selbst bleibt unverändert! Beachten Sie bitte, dass nur die Änderungen dieser Revision rückgängig gemacht werden. Es wird nicht die gesamte Datei in der früheren Revision wieder hergestellt. Dieser Befehl ist nützlich, wenn Sie Änderungen zurücknehmen wollen, aber spätere, nicht damit zusammenhängende Änderungen beibehalten möchten.

Falls Ihre Arbeitskopie vor dieser Aktion unmodifiziert war, wird sie anschließend das modifiziert Symbol anzeigen. Falls Sie bereits lokale Änderungen vorgenommen haben, wird dieser Befehl die *Rückgängig*-Änderungen in Ihrer Arbeitskopie zusammenführen.

Intern führt Subversion die Änderungen dieser Revision rückwärts zusammen, so dass sie aus einer vorherigen Übertragung entfernt werden.

Sie können, wie unter *Rückgängig* zu dieser Revision beschrieben, die *Rückgängig Aktion rückgängig machen*.

Revision zusammenführen in...

Führt die gewählten Revisionen in einer anderen Arbeitskopie zusammen. Ein Dialog ermöglicht es Ihnen, die Ziel-Arbeitskopie auszuwählen, aber danach gibt es weder eine Bestätigung noch eine Möglichkeit, einen Testlauf durchzuführen. Es ist eine gute Idee, das Zusammenführen in einer unmodifizierten Arbeitskopie zu testen, so dass Sie die Änderungen rückgängig machen können, wenn es nicht geklappt hat! Dies ist eine nützliche Funktion, wenn Sie Revisionen von einem Zweig auf einen anderen übertragen möchten.

Auschecken...

Eine neue Arbeitskopie des markierten Ordners in der gewählten Revision erstellen. Im folgenden Dialog können Sie die URL und Revision überprüfen und ein Ziel für die Arbeitskopie festlegen.

Exportieren...

Den markierten Ordner in der gewählten Revision exportieren. Im folgenden Dialog können Sie die URL und Revision überprüfen und ein Ziel für den Export festlegen.

Autor/Logmeldung bearbeiten

Die Logmeldung oder den Autor der Übertragung ändern. Lesen Sie [Abschnitt 4.9.7, „Ändern der Logmeldung und des Autors“](#), um herauszufinden wie dies funktioniert.

Zeige Revisionseigenschaften

Eine beliebige Revisionseigenschaft, nicht nur Logmeldung und Autor, bearbeiten. Weiteres in [Abschnitt 4.9.7, „Ändern der Logmeldung und des Autors“](#).

In die Zwischenablage kopieren

Die Logdetails der gewählten Revisionen in die Zwischenablage kopieren. Dieser Befehl kopiert Revisionsnummer, Autor, Datum, Logmeldung sowie die Liste der geänderten Objekte jeder Revision in die Zwischenablage.

Logmeldungen durchsuchen...

Die Logmeldungen durchsuchen. Die von Subversion generierte Liste der geänderten Dateien wird einbezogen. Die Suche ignoriert Groß-/Kleinschreibung.

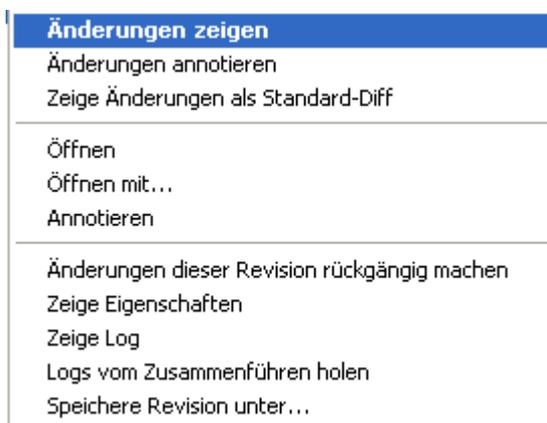


Abbildung 4.17. Kontextmenü des Log-Dialogs für zwei ausgewählte Revisionen

Wenn Sie (mit **Strg**) zwei Revisionen auf einmal auswählen, reduziert sich das Kontextmenü auf folgende Funktionen:

Revisionen vergleichen

Die gewählten Revisionen vergleichen. Das Standard Vergleichsprogramm ist TortoiseMerge, welches mit TortoiseSVN installiert wird.

Wenn Sie diese Funktion für einen Ordner wählen, erscheint ein Folgedialog, der die geänderten Dateien anzeigt und Ihnen weitere Vergleichsoptionen anbietet. Lesen Sie mehr dazu in [Abschnitt 4.10.3, „Ordner vergleichen“](#).

Revisionen annotieren

Die gewählten Revisionen annotieren und die Ergebnisse mit einem Vergleichsprogramm darstellen. Lesen Sie [Abschnitt 4.23.2, „Unterschiede annotieren“](#) für weitere Details.

Unterschiede als Standard-Diff anzeigen

Die gewählten Revisionen vergleichen und als Standard-Diff anzeigen. Diese Funktion steht sowohl für Dateien, als auch für Verzeichnisse zur Verfügung.

In die Zwischenablage kopieren

Die Logmeldungen, wie oben beschrieben, in die Zwischenablage kopieren.

Logmeldungen durchsuchen...

Die Logmeldungen durchsuchen. Näheres dazu siehe oben.

Wenn Sie mehrere aufeinander folgende Revisionen (mittels der üblichen **Strg** oder **Umsch** Tasten) markieren, bietet das Kontextmenü eine Funktion an, alle Änderungen innerhalb des Revisionsbereiches rückgängig zu machen. Dies ist der einfachste Weg, einen Satz von Änderungen in einem Schritt zurückzunehmen.

Sie können, wie bereits oben beschrieben, die gewählten Revisionen auch in einer anderen Arbeitskopie zusammenführen.

Falls alle markierten Revisionen den selben Autor haben, können Sie den Autor dieser Revisionen in einem Schritt ändern.

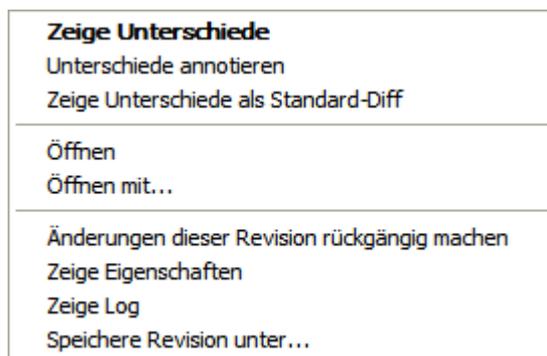


Abbildung 4.18. Kontextmenü der Dateiliste des Log-Dialogs

Die Liste der veränderten Dateien besitzt ebenfalls ein Kontextmenü mit folgenden Funktionen:

Änderungen anzeigen

Die Änderungen der markierten Datei in der gewählten Revision anzeigen. Nur für Dateien verfügbar, die als *Verändert* gekennzeichnet sind.

Änderungen annotieren

Die markierte sowie die vorherige Revision der gewählten Datei annotieren und Ergebnisse mit einem Vergleichsprogramm darstellen. Lesen Sie [Abschnitt 4.23.2, „Unterschiede annotieren“](#) für weitere Informationen.

Zeige als Standard-Diff

Die Änderungen der markierten Datei als Standard-Diff anzeigen. Nur für Dateien verfügbar, die als *Verändert* gekennzeichnet sind.

Öffnen / Öffnen mit...

Die gewählte Datei entweder mit dem Standardprogramm für den Dateityp oder mit einem von Ihnen gewählten Programm öffnen.

Annotieren...

Öffnet den Annotieren-Dialog, mit dessen Hilfe Sie bis zur markierten Revision annotieren können

Änderungen dieser Revision rückgängig machen

Die Änderungen der gewählten Datei in dieser Revision rückgängig machen.

Zeige Eigenschaften

Die Subversion-Eigenschaften für das ausgewählte Objekt anzeigen.

Zeige Log

Die Logmeldungen für die ausgewählte Datei anzeigen.

Integrationslogs anzeigen

Zeigt das Revisionslog inklusive zusammengeführten Änderungen für eine einzelne markierte Datei. Weiteres unter [Abschnitt 4.9.6, „Datenintegration protokollieren“](#).

Speichere Revision unter...

Die gewählte Revision speichern, so dass Sie eine ältere Version der Datei erhalten.



Tipp

Sie werden bemerkt haben, dass wir manchmal von Änderungen und manchmal von Unterschieden sprechen. Worin liegt der Unterschied?

Subversion verwendet Revisionsnummern in zwei verschiedenen Zusammenhängen. Zum Einen referenziert eine Revisionsnummer den Status eines Projektarchives zu einem gewissen Zeitpunkt. Zum Anderen wird damit auch der Satz von Änderungen bezeichnet, der diese Revision erzeugt hat. „Erledigt in Revision r1234“ bedeutet z.B. das Funktion X in Revision 1234 implementiert wurde. Um die Bedeutung klarer zu machen, verwenden wir zwei verschiedene Begriffe.

Wenn Sie zwei Revisionen N und M wählen, wird das Kontextmenü Ihnen die *Unterschiede* zwischen diesen Revisionen anbieten. In Subversion Terminologie also `diff -r M:N`.

Wenn Sie eine einzelne Revision N wählen, wird das Kontextmenü Ihnen die *Änderungen* in dieser Revision anbieten. In Subversion Terminologie also `diff -r N-1:N` oder `diff -c N`.

Der untere Bereich zeigt die in allen gewählten Revisionen veränderten Dateien, so dass über das Kontextmenü stets *Änderungen* angezeigt werden können.

4.9.4. Weitere Logmeldungen holen

Der Log-Dialog zeigt aus mehreren Gründen nicht immer alle Änderungen die jemals gemacht wurden an:

- Bei einem großen Projektarchiv können hunderte oder gar tausende von Änderungen gespeichert sein. Alle diese Änderungen auf einmal zu laden kann sehr lange dauern. Normalerweise ist man auch nur an den letzten Meldungen interessiert. Standardmäßig werden beim ersten Aufruf des Log-Dialogs die letzten einhundert Logmeldungen angezeigt. Dieser Wert kann im Einstellungs-Dialog geändert werden. TortoiseSVN → Einstellungen ([Abschnitt 4.30.1.2, „TortoiseSVN Dialoge Seite 1“](#)),

- Wenn die Option **Bei Kopien/Umbenennen anhalten** gewählt ist, werden keine weiteren Logmeldungen mehr geholt, falls der Ordner / die Datei von einer anderen Stelle des Projektarchivs kopiert wurde. Dies ist insbesondere dann von Nutzen, wenn man Zweige oder Marken betrachtet und nur die Änderungen innerhalb des Zweiges sehen möchte.

Normalerweise werden Sie diese Option nicht aktivieren. TortoiseSVN merkt sich die von Ihnen gewählte Einstellung.

Wenn der Log-Dialog aus dem Zusammenführen-Dialog heraus aufgerufen wird, ist diese Option standardmäßig aktiviert. Der Grund dafür ist, dass beim Zusammenführen meistens Änderungen an Zweigen betrachtet werden. Es ist normalerweise nicht sinnvoll, über die Wurzel des Zweiges hinaus in die Vergangenheit zu gehen.

Beachten Sie, dass Subversion Umbenennen derzeit als aufeinanderfolgende Kopieren/Löschen Aktionen implementiert, so dass die Log Anzeige ebenfalls anhält, wenn ein Ordner / eine Datei umbenannt wurde, falls diese Option gewählt ist.

Wenn Sie mehr Logmeldungen sehen wollen, klicken Sie auf die **Nächste 100** Schaltfläche um weiter 100 Meldungen zu holen. Sie können dies so oft wie nötig wiederholen.

Daneben befindet sich eine Mehrfachschaltfläche, die sich den letzten gewählten Zustand merkt. Klicken Sie auf den Pfeil, um die anderen Möglichkeiten zu sehen.

Wählen Sie **Zeige Bereich**, wenn Sie einen bestimmten Revisionsbereich betrachten wollen. Ein Dialog wird dann die Anfangs- und Endrevision von Ihnen abfragen.

Wählen Sie **Zeige Alle**, wenn Sie *alle* Logmeldungen von HEAD zurück bis zur Revision 1 sehen möchten.

4.9.5. Aktuelle Revision der Arbeitskopie

Da der Log-Dialog Ihnen die Logmeldungen ab der HEAD Revision zeigt, kann es vorkommen, dass Logmeldungen für Änderungen angezeigt werden, die sich noch nicht in Ihrer Arbeitskopie befinden. Um dies zu verdeutlichen, wird die Logmeldung, die der aktuellen Revision der Arbeitskopie entspricht in Fettdruck hervorgehoben.

Wenn Sie das Log für einen Ordner anzeigen, entspricht die hervorgehobene Revision der höchsten, innerhalb des Ordners gefundenen Revisionsnummer. Diese zu ermitteln erfordert ein rekursives Durchsuchen der Unterordner. Dies kann für große Arbeitskopien eine langwierige Operation sein. Wenn Sie diese Funktion deaktivieren oder einschränken wollen, müssen Sie einen Registrierungsschlüssel in `HKCU\Software\TortoiseSVN\RecursiveLogRev` entsprechend der Beschreibung in [Abschnitt 4.30.10, „Einstellungen in der Registrierung“](#) anlegen.

4.9.6. Datenintegration protokollieren

Subversion 1.5 und neuer speichern die in einem Zweig zusammengeführten Revisionen in einer Eigenschaft des Zweiges ab. Dies ermöglicht es, eine detaillierte Historie von bereits durchgeführten Integrationsschritten zu erhalten. Wenn Sie zum Beispiel eine neue Funktion in einem Zweig entwickeln und die fertige Entwicklung im Stamm zusammenführen, wird das Log für den Stamm nur eine Revision anzeigen, obwohl darin vielleicht 1000 Revisionen des Zweiges enthalten sind.

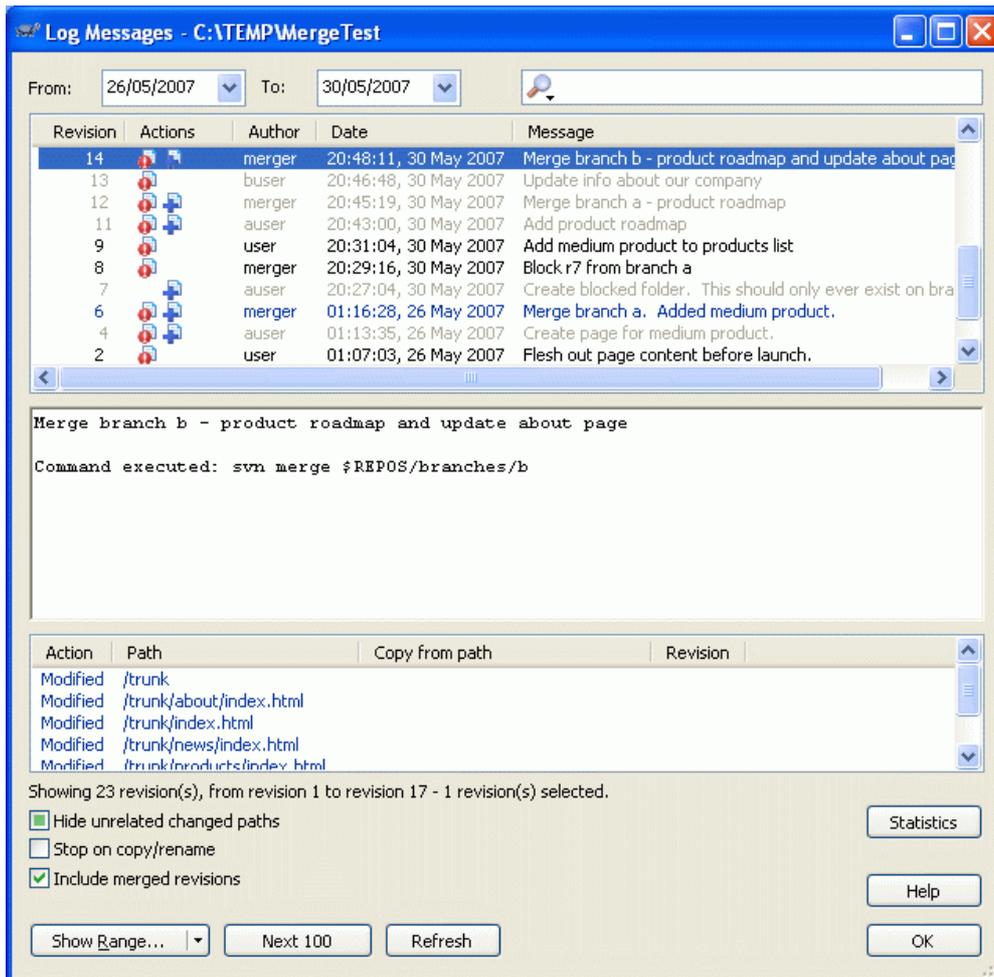


Abbildung 4.19. Der Log-Dialog mit bereits zusammengeführten Revisionen

Wenn Sie im Detail sehen wollen, welche Revisionen als Teil einer Übertragung integriert wurden, aktivieren Sie die Option **Versionen vom Zusammenführen einschließen**. Damit werden die Logmeldungen erneut geholt und die Einträge aus den zusammengeführten Revisionen dazwischen sortiert. Zusammengeführte Revisionen werden in grau angezeigt, weil es sich um Änderungen an einem anderen Teil des Baumes handelt.

Datenintegration ist niemals einfach! Während Sie eine Funktion auf dem Zweig entwickeln, werden Sie dort sicher ab und zu Änderungen aus dem Stamm zusammenführen, um mit der Hauptentwicklung synchron zu bleiben. Deshalb wird die Historie des Zweiges eine weitere Ebene an Informationen über Datenintegration enthalten. Diese verschiedenen Ebenen werden durch unterschiedlich tiefe Einrückungen im Log-Dialog angezeigt.

4.9.7. Ändern der Logmeldung und des Autors

Revisionseigenschaften unterscheiden sich vollständig von den Subversion Eigenschaften jedes Objekts. Revisionseigenschaften sind, mit einer bestimmten Revisionsnummer des Projektarchivs verbundene, beschreibende Objekte, zum Beispiel Logmeldung, Datum und Autor der Übertragung.

Manchmal ist es notwendig, eine Logmeldung nachträglich zu ändern. Sei es weil die Meldung Rechtschreibfehler enthält oder weil Sie die Meldung mit mehr Informationen erweitern müssen oder aus anderen Gründen Oder Sie müssen den Autor der Revision ändern, weil Sie vielleicht vergessen haben eine Authentifizierung für Übertragungen auf dem Server einzurichten oder...

Subversion lässt Sie Revisionseigenschaften jederzeit ändern. Aber da solche Änderungen nicht wieder rückgängig gemacht werden können (diese Änderungen werden nicht aufgezeichnet) ist die Funktion standardmäßig deaktiviert. Um diese Funktion zu aktivieren, müssen Sie eine `pre-revprop-change` Aktion für das Projektarchiv einrichten. Bitte lesen Sie dazu im Kapitel *Hook Scripts* [<http://svnbook.red-bean.com/en/1.5/svn.reposadmin.create.html#svn.reposadmin.create.hooks>] des Subversion Buchs nach, wie das gemacht wird. Lesen Sie auch **Abschnitt 3.3, „Serverseitige Aktionsskripte“**, um mehr Informationen über das Einrichten von Aktionen auf einem Windows PC zu erhalten.

Sobald Sie die gewünschten Aktionensskripte für das Projektarchiv eingerichtet haben, können Sie sowohl den Autor als auch die Logmeldung (oder jede andere Revisionseigenschaft) über das Kontextmenü der Revisionsliste nachträglich ändern. Die Logmeldung selbst kann auch über das Kontextmenü des Textfeldes geändert werden.



Warnung

Da die Revisionseigenschaften (z.B. `svn:log` für Logmeldungen) in Subversion nicht aufgezeichnet werden, überschreiben Änderungen den alten Wert für *immer*.

4.9.8. Logmeldungen filtern

Mit Hilfe der Filter am oberen Rand des Log-Dialogs können Sie die Logmeldungen auf diejenigen einschränken, die Sie interessieren. Die Von und Bis ermöglichen es, die Ausgabe auf einen bestimmten Zeitraum zu reduzieren. Die Suchmaske filtert die Meldungen heraus, die die angegebene Zeichenfolge beinhalten

Klicken Sie auf das Suchsymbol, um auszuwählen welche Werte Sie durchsuchen möchten und um die Suche mit regulären Ausdrücken zu aktivieren. Normalerweise benötigen Sie nur eine einfache Textsuche, aber falls Sie komplexere Suchbegriffe formulieren wollen, können Sie reguläre Ausdrücke verwenden. Wenn Sie mit dem Mauszeiger über das Eingabefeld fahren, erscheint ein Hinweistext zu den regulären Ausdrücken. Unter <http://www.regular-expressions.info/> stehen eine Dokumentation und Übungen zur Verfügung. Der Filter vergleicht Ihren Suchtext mit den Logmeldungen und zeigt nur die Einträge an, die mit dem Suchtext *übereinstimmen*.

Damit der Filter alle Einträge anzeigt, die *nicht* mit dem Suchtext übereinstimmen, beginnen Sie den Suchtext mit einem Ausrufezeichen (!). Zum Beispiel würde der Suchtext `!Harry` nur die Einträge anzeigen, die nicht durch `Harry` übertragen wurden.

Beachten Sie dass diese Filter nur auf die bereits heruntergeladenen Meldungen wirken. Sie haben keinen Einfluss darauf, welche Meldungen aus dem Projektarchiv heruntergeladen werden.

Sie können auch die Pfade in der unteren Liste filtern indem sie die **Geänderte Pfade ohne Bezug ausblenden** Option nutzen. Pfade mit Bezug sind Pfade welche den Pfad für welchen sie den Log anzeigen drin haben. Wenn sie den Log für einen Ordner anzeigen heißt dies dass alle Pfade unterhalb dieses Ordners einen Bezug haben, alle anderen Pfade nicht. Bei Dateien hat nur die Datei selbst einen Bezug mit sich selbst. Das Optionsfeld hat drei Zustände: sie können entweder alle Pfade anzeigen, Pfade ohne Bezug grau anzeigen oder ganz ausblenden.

Manchmal erfordern es Ihre Arbeitsgewohnheiten, dass Logmeldungen ein bestimmtes Format haben. Dadurch ist unter Umständen der Text, der die Änderungen beschreibt in der Zusammenfassung von TortoiseSVN nicht sichtbar. Die Eigenschaft `tsvn:logsummary` kann dazu verwendet werden, einen Teil der Logmeldung zu extrahieren und als Zusammenfassung anzuzeigen. In **Abschnitt 4.17.2, „TortoiseSVN Projekteigenschaften“** wird beschrieben, wie die Eigenschaft benutzt wird.



Keine Log-Formatierung im Projektarchivbetrachter

Da die Formatierung vom Zugriff auf Subversion Eigenschaften abhängt, sehen Sie die Ergebnisse nur in einer Arbeitskopie. Im Projektarchivbetrachter steht diese Funktion nicht

zur Verfügung, weil das Holen der Eigenschaften von einem Server eine zeitintensive Operation list.

4.9.9. Statistiken anzeigen

Die Statistiken Schaltfläche zeigt ein Formular mit interessanten Informationen über die angezeigten Revisionen an. Es zeigt zum Beispiel wie viele Autoren (Personen welche Übertragungen gemacht haben) am Werk waren, wie viele Übertragungen jeder von ihnen gemacht hat, wöchentlicher Projektfortschritt und vieles mehr. Endlich haben Sie die Möglichkeit zu überprüfen, wer fleißig am Projekt mitarbeitet und wer hinterherhängt ;-)

4.9.9.1. Statistik Seite

Diese Seite zeigt Ihnen sämtliche vorstellbaren Zahlen an. Insbesondere der Datumsbereich und die zugehörigen Revisionen sowie einige Min-/Max-/Mittelwerte.

4.9.9.2. Übertragungen per Autor

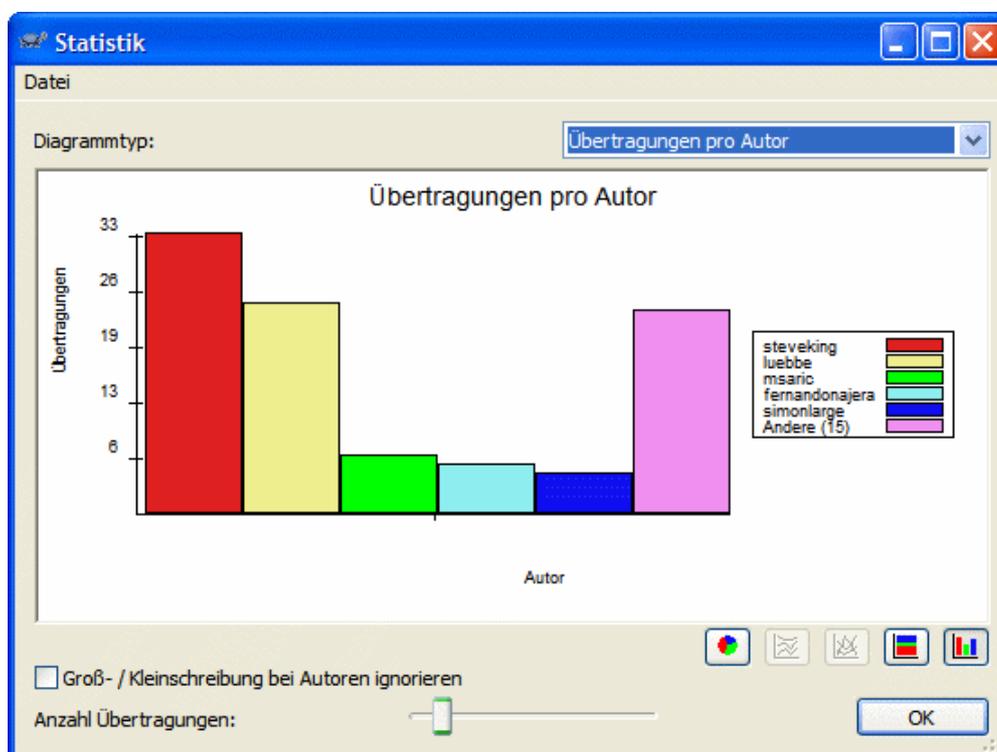


Abbildung 4.20. Übertragungen per Autor als Histogramm

Dieser Graph zeigt welche Autoren aktiv am Projekt gearbeitet haben als einfaches Histogramm, überlagertes Histogramm oder Kuchendiagramm.

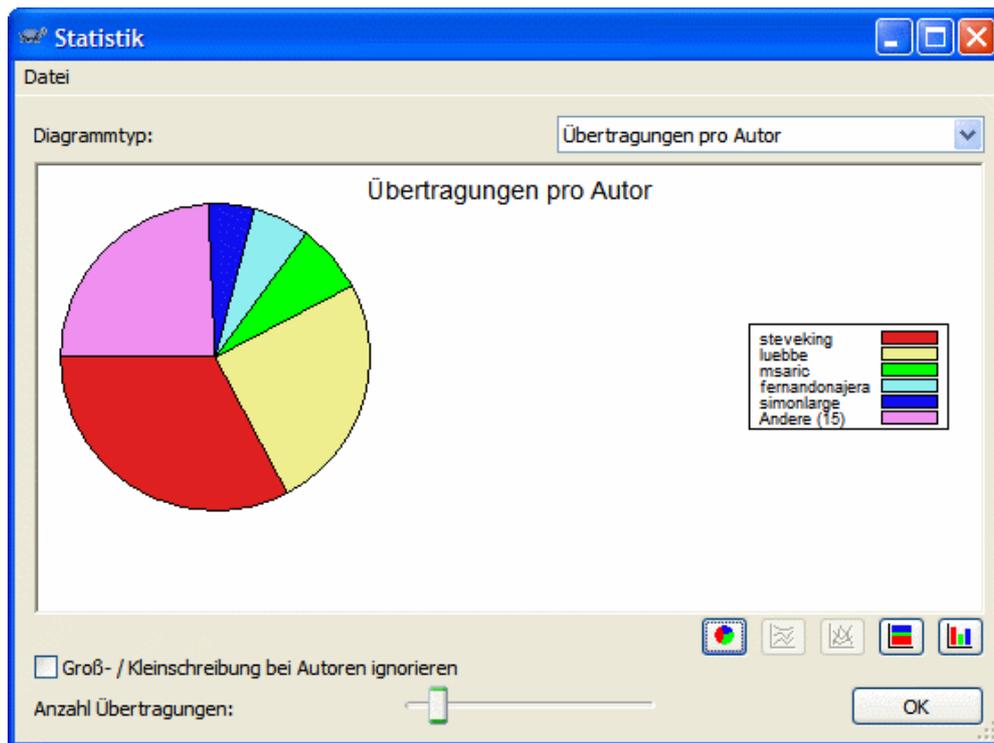


Abbildung 4.21. Übertragungen per Autor als Tortendiagramm

Wenn es einige wenige Hauptautoren und viele Autoren mit wenigen Beiträgen gibt, kann die Anzahl der kleinen Segmente den Graphen unleserlich machen. Der Schieber am unteren Rand erlaubt es Ihnen, einen Prozentwert der gesamten Übertragungen als Schwellwert festzulegen, unterhalb dessen die Autoren in der Gruppe *Andere* zusammengefasst werden.

4.9.9.3. Die Seite Übertragungen nach Datum

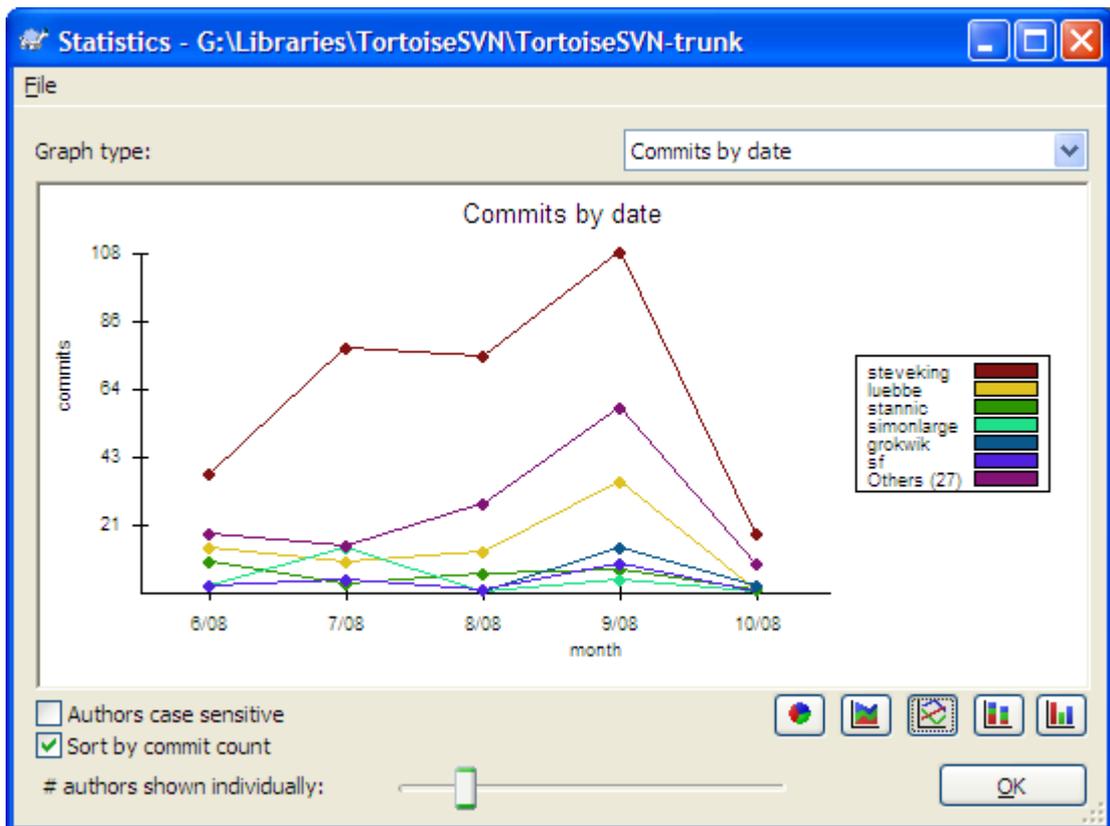


Abbildung 4.22. Übertragungen nach Datum

Diese Seite zeigt Ihnen eine Zusammenfassung der Projektaktivität in Hinblick auf Übertragungen *und* Autoren an. Damit können Sie auf einen Blick erkennen, wann und durch wen am Projekt gearbeitet wurde.

Sobald es mehrere Autoren gibt, werden Sie entsprechend viele Linien im Diagramm sehen. Es gibt zwei verschiedene Ansichten: *normal*, in der die Aktivität eines Autors relativ zur Grundlinie angezeigt wird und *gestapelt*, bei der die Aktivität eines Autors relativ zur vorherigen Linie angezeigt wird. Die zweite Option vermeidet sich kreuzende Linien, erschwert es aber auch, die Aktivität eines einzelnen Autors zu bewerten.

Standardmäßig wird Groß-/Kleinschreibung bei der Analyse berücksichtigt, so dass PeterEgan und PeteRegan als zwei verschiedene Autoren erkannt werden. In vielen Fällen werden Benutzernamen inkonsistent eingegeben, Deshalb können Sie den Vergleich mit Groß-/Kleinschreibung bei Autoren ignorieren entsprechend anpassen.

Beachten Sie, dass die Statistiken dieselbe Zeitperiode überdecken wie Revisionen im Log-Dialog angezeigt werden. Wenn also nur eine Revision angezeigt wird, sagen die Statistiken nicht sehr viel aus.

4.9.10. Offline Modus

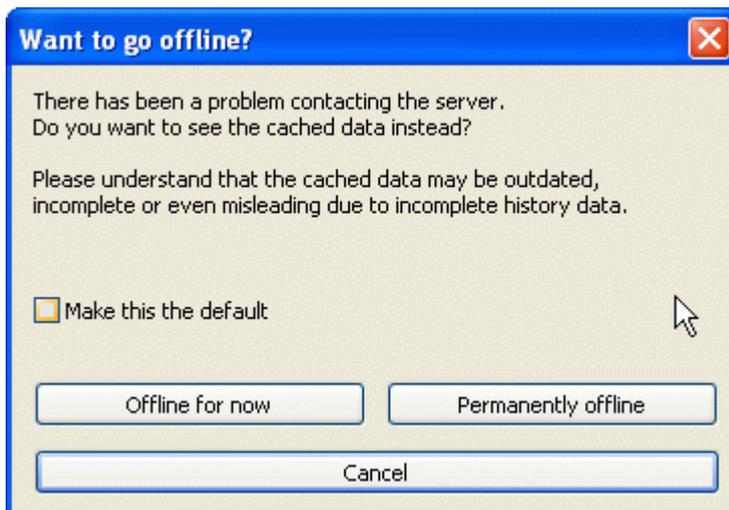


Abbildung 4.23. Offline gehen

Wenn der Server nicht erreichbar ist und Sie den Log-Puffer aktiviert haben, können Sie den Log-Dialog und den Revisionsgraphen auch offline benutzen. Dabei werden die Daten aus dem Puffer gelesen, so dass Sie wie gewohnt weiterarbeiten können. Allerdings ist es möglich, dass die angezeigten Informationen veraltet oder unvollständig sind.

Hier haben Sie drei Möglichkeiten:

Für diese Aktion offline

Beendet die aktuelle Operation im Offline-Modus, versucht aber, wenn das nächste Mal Logdaten angefordert werden, wieder auf das Projektarchiv zuzugreifen.

Permanent offline

Bleibt im Offline-Modus bis eine Aktualisierung der Logdaten explizit angefordert wird. Siehe [Abschnitt 4.9.11, „Die Ansicht aktualisieren“](#).

Abbrechen

Wenn Sie die Operation wegen möglicherweise veralteter Logdaten nicht fortsetzen wollen, brechen Sie einfach ab.

Die Option *Diese Einstellung immer verwenden* verhindert, dass der Dialog wieder erscheint und verwendet stets die von Ihnen gewählte Aktion. Sie können die Vorgabe jederzeit über TortoiseSVN → Optionen ändern.

4.9.11. Die Ansicht aktualisieren

Wenn Sie den Server auf neue Logmeldungen überprüfen wollen, können Sie die Sicht einfach per **F5** aktualisieren. Falls Sie den Log-Puffer verwenden (standardmäßig aktiv), wird das Projektarchiv auf neue Meldungen abgefragt und nur diese werden geholt. Falls der Log-Puffer offline war, wird gleichzeitig versucht, ihn wieder online zu schalten.

Falls Sie den Log-Puffer verwenden und denken, dass sich der Inhalt oder der Autor einer Logmeldung geändert haben, können Sie **Umsch-F5** oder **Strg-F5** benutzen, um die angezeigten Meldungen erneut vom Server zu holen. Beachten Sie, dass das nur die gerade angezeigten Meldungen betrifft und dass nicht der gesamte Puffer für das Projektarchiv ungültig wird.

4.10. Unterschiede anzeigen

Eine der häufigsten Anforderungen in der Projektarbeit ist herauszufinden, was sich geändert hat. Entweder möchten Sie die Unterschiede zwischen zwei Revisionen derselben Datei sehen oder aber die Unterschiede zwischen zwei verschiedenen Dateien. TortoiseSVN bietet Ihnen mit TortoiseMerge ein Programm an, das die meisten Anforderungen erfüllt, erlaubt es ihnen aber auch, ein eigenes Programm zum Anzeigen und Bearbeiten von Differenzen zu verwenden. Um Unterschiede von Bilddateien zu erkennen bietet TortoiseSVN ebenfalls ein Programm namens TortoiseIDiff an.

4.10.1. Datei-Unterschiede

Lokale Änderungen

Wenn Sie sehen wollen, welche Änderungen *Sie selbst* in Ihrer Arbeitskopie haben, verwenden Sie das Kontextmenü des Explorers und wählen Sie TortoiseSVN → Vergleiche.

Unterschiede zu einem anderen Zweig

Wenn Sie sehen wollen, was sich im Stamm (wenn Sie auf einem Zweig arbeiten) oder in einem bestimmten Zweig (wenn Sie am Stamm arbeiten) geändert hat, können Sie das Explorer Kontextmenü verwenden. Halten Sie die **Umsch** Taste gedrückt, wählen Sie einen Rechtsklick auf die Datei machen. Dann wählen Sie TortoiseSVN → Vergleich mit URL. Im folgenden Dialog geben Sie die URL im Projektarchiv an, mit der Sie die lokale Datei vergleichen wollen.

Sie können auch den Projektarchivbetrachter zum Vergleich zweier Bäume verwenden, indem Sie zwei Marken oder auch einen Zweig und den Stamm auswählen. Das Kontextmenü bietet Ihnen dann die Option an, diese Bäume zu vergleichen Compare revisions. Lesen Sie weiter in [Abschnitt 4.10.3, „Ordner vergleichen“](#).

Unterschiede zu einer früheren Revision

Wenn Sie die Unterschiede zwischen einer bestimmten Revision und Ihrer Arbeitskopie sehen wollen, lassen Sie sich den Log-Dialog anzeigen, markieren dort die gewünschte Revision und wählen Vergleiche mit Arbeitskopie aus dem Kontextmenü.

Wenn Sie den Unterschied zwischen der letzten übertragenen Revision und Ihrer Arbeitskopie, unter der Annahme, dass die Arbeitskopie noch nicht verändert wurde, sehen wollen, machen Sie einen Rechtsklick auf die Datei. Dann wählen Sie TortoiseSVN → Vergleiche mit vorheriger Revision. Dadurch führen Sie einen Vergleich mit der Revision vor dem letzten, in Ihrer Arbeitskopie gespeicherten, Übertragungszeitpunkt und der Arbeitskopie BASE Revision durch. Dieser Vergleich reflektiert die Änderungen, durch die Ihre Arbeitskopie aktualisiert wurde. Sie sehen keine Änderungen neueren Datums.

Unterschiede zwischen zwei vorherigen Revisionen

Wenn Sie die Unterschiede zwischen zwei bereits übertragenen Revisionen sehen wollen, öffnen Sie den Log-Dialog und markieren Sie die beiden Revisionen mit gedrückter **Strg** Taste. Dann wählen Sie Revisionen vergleichen aus dem Kontextmenü.

Wenn Sie diese Option für das Log eines Ordners wählen, erscheint ein weiterer Dialog, der die geänderten Dateien anzeigt und Ihnen weitere Vergleichsoptionen anbietet. Lesen Sie mehr dazu in [Abschnitt 4.10.3, „Ordner vergleichen“](#).

Alle Änderungen einer Übertragung

Wenn Sie die Änderungen in allen Dateien in einer bestimmten Revision auf einen Blick sehen wollen, verwenden Sie das Standard-Diff (GNU Patch). Dieses Format zeigt nur die Unterschiede mit ein paar Zeilen Kontext an. Es ist schwieriger zu lesen als ein visueller Vergleich, aber es zeigt alle Änderungen auf einmal. Markieren Sie dazu im Log-Dialog die gewünschte Revision und wählen Sie Zeige Unterschiede als Standard-Diff aus dem Kontextmenü.

Unterschiede zwischen Dateien

Wenn Sie die Unterschiede zwischen zwei Dateien sehen wollen, können Sie das direkt im Explorer tun, indem Sie die beiden Dateien markieren und aus dem Explorer Kontextmenü TortoiseSVN → Vergleiche wählen.

Unterschiede zwischen Arbeitskopie Datei/Ordner und URL

Wenn Sie die Unterschiede zwischen einer Datei in der Arbeitskopie und einer Datei im Subversion Projektarchiv sehen möchten, können sie das erreichen, indem Sie die **Umsch** Taste drücken, während sie einen Rechtsklick auf die Datei machen. Wählen sie dann TortoiseSVN → **Vergleichen mit URL**. Sie können diese Aktion ebenso für einen Ordner in der Arbeitskopie durchführen. TortoiseMerge wird Ihnen die Unterschiede wie eine Patchdatei anzeigen - eine Liste von Dateien, die Sie einzeln betrachten können.

Unterschied mit Annotierungsinformation

Wenn Sie nicht nur die Unterschiede, sondern auch Autor, Revision und Datum der Änderung sehen wollen, können Sie Vergleichen und Annotieren über den Log-Dialog miteinander kombinieren. Lesen Sie [Abschnitt 4.23.2, „Unterschiede annotieren“](#) für weitere Informationen.

Unterschiede zwischen Ordnern

Die mit TortoiseSVN gelieferten Programme unterstützen keinen Vergleich von Verzeichnissen. Wenn Sie über ein externes Programm verfügen, das diese Funktionalität bietet, können Sie das stattdessen benutzen. In [Abschnitt 4.10.5, „Externe Programme“](#) stellen wir Ihnen einige Programme vor, die wir mit TortoiseSVN getestet haben.

Wenn Sie ein externes Vergleichsprogramm konfiguriert haben, können Sie die **Umsch**-Taste beim Vergleichen drücken, um das alternative Vergleichsprogramm aufzurufen. Lesen Sie in [Abschnitt 4.30.5, „Einstellungen für externe Programme“](#) nach, wie weitere Vergleichsprogramme eingerichtet werden.

4.10.2. Zeilenende- und Leerzeichenoptionen

Manchmal werden Sie während eines Projektes die Zeilenenden von CRLF auf LF umstellen oder Sie werden die Einrückung eines Abschnitts ändern. Unglücklicherweise wird dadurch eine große Anzahl Zeilen als verändert markiert obwohl diese Unterschiede die Bedeutung des Programmcodes nicht beeinflussen. Mit den folgenden Optionen können Sie einstellen, wie diese Unterschiede beim Vergleichen behandelt werden sollen. Sie finden die Einstellungen in den Zusammenführen- und Annotieren-Dialogen sowie in den Einstellungen für TortoiseMerge.

Ignoriere Zeilenende schließt Änderungen aus, die nur aus unterschiedlichen Zeilenumbrüchen bestehen.

Vergleiche Leerzeichen schließt Änderungen in Leerzeichen und Einrückung als gelöschte / hinzugefügte Zeilen ein.

Ignoriere Änderungen in Leerzeichen schließt Änderungen in der Anzahl oder dem Typ von Leerzeichen, z.B. Änderung der Einrückung oder Wechsel von Tab zu Leerzeichen aus. Komplett neue oder vollständig gelöschte Leerzeichen werden weiterhin als Änderung angezeigt.

Ignoriere alle Leerzeichen schließt sämtliche Änderungen, die nur aus Leerzeichen bestehen, aus.

Selbstverständlich wird jede Zeile mit geänderten Inhalten stets als Differenz angezeigt.

4.10.3. Ordner vergleichen

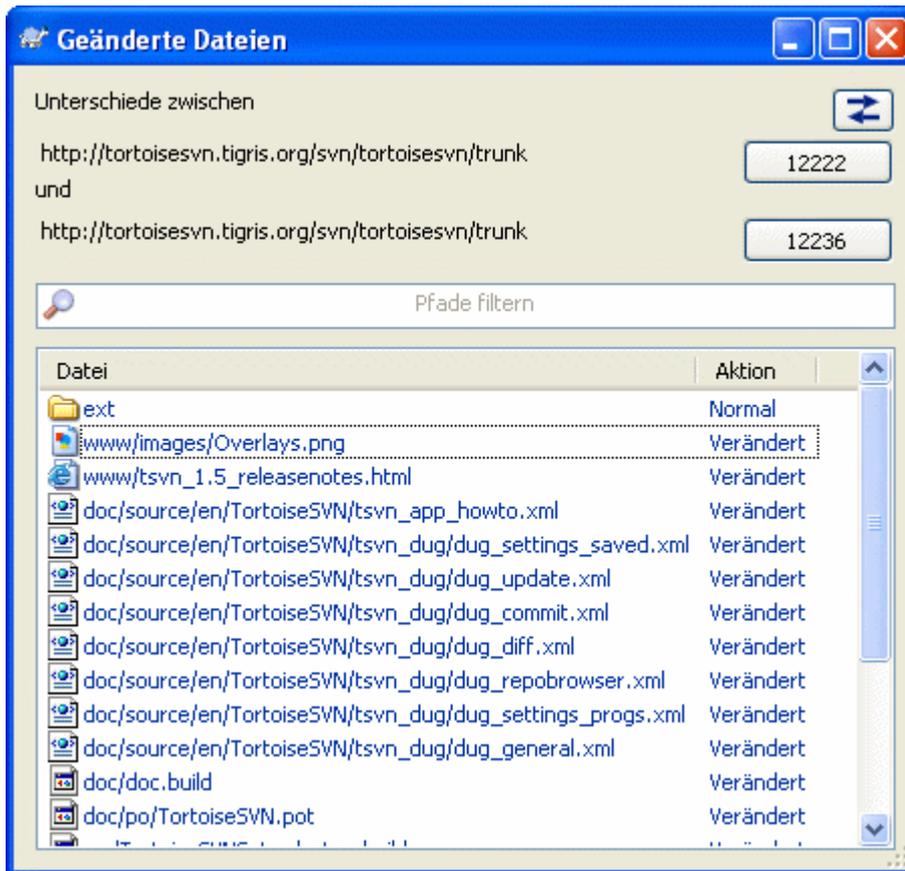


Abbildung 4.24. Der Dialog zum Vergleichen von Revisionen

Wenn Sie zwei Bäume im Projektarchivbetrachter markieren oder wenn Sie zwei Revisionen einer Datei im Logdialog wählen, können Sie Kontextmenü → Revisionen vergleichen aufrufen.

Dieser Dialog zeigt eine Liste aller Dateien, die geändert wurden und erlaubt Ihnen, sie mit Hilfe des Kontextmenüs einzeln zu Vergleichen oder zu annotieren.

Sie können einen *Änderungsbaum* exportieren. Das ist nützlich, wenn Sie jemandem Ihre Projektstruktur schicken wollen, die aber nur die geänderten Dateien enthalten soll. Diese Funktion arbeitet nur auf den markierten Dateien. Das bedeutet, dass Sie die gewünschten Dateien - normalerweise alle - markieren und dann per Kontextmenü → Exportiere Auswahl in... in den im folgenden Dialog abgefragten Ort herauschreiben.

Sie können auch die *Liste* der geänderten Dateien in eine Textdatei herauschreiben, indem Sie Kontextmenü → Speichere Liste der gewählten Dateien in... wählen.

Wenn Sie die Liste der Dateien *und* der Aktionen (verändert, hinzugefügt, gelöscht) exportieren wollen, können Sie dies Kontextmenü → Auswahl in Zwischenablage kopieren erreichen.

Die obere Schaltfläche ermöglicht es Ihnen, die Richtung des Vergleichs festzulegen. Sie können sich entweder die Änderungen anzeigen lassen, um von A nach B zu kommen oder umgekehrt.

Die Schaltflächen mit den Revisionsnummern dienen dazu, einen anderen Revisionsbereich auszuwählen. Wenn Sie den Bereich ändern, wird die Liste der Objekte, die sich darin geändert haben, aktualisiert.

Wenn die Liste der Dateien sehr lang ist, kann sie mit Hilfe des Suchfeldes auf die Dateien mit einer bestimmten Zeichenfolge beschränkt werden. Es wird eine *einfache Textsuche* durchgeführt. Wenn Sie z.B. nur C Quelltexte angezeigt haben wollen, geben Sie `.c` anstatt `*.c` ein.

4.10.4. Bilder mit TortoiseIDiff vergleichen

Es gibt viele Programme, einschließlich unseres TortoiseMerge, mit denen Unterschiede zwischen Textdateien angezeigt werden können. Wir haben uns aber schon des öfteren zu sehen gewünscht, wie sich ein Bild geändert hat. Deshalb haben wir TortoiseIDiff entwickelt.

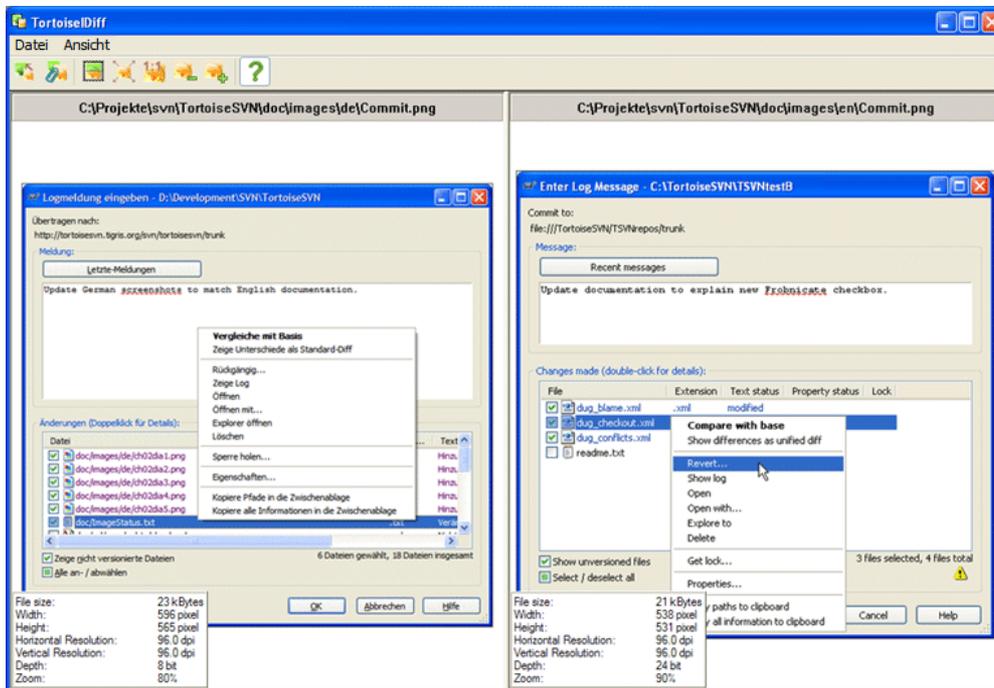


Abbildung 4.25. Ein Programm zum Vergleichen von Bildern

TortoiseSVN → Vergleich für jedes übliche Bildformat, wird TortoiseIDiff starten, um die Unterschiede anzuzeigen. In der Standardansicht werden die Bilder nebeneinander dargestellt. Mit der Symbolleiste oder dem Menü können Sie die Bilder auch übereinander darstellen beziehungsweise überlagern. Der Schieber ermöglicht es Ihnen, die Transparenz der Überlagerung einstellen.

Außerdem können Sie die Ansicht vergrößern oder verkleinern und den aktuellen Bildausschnitt mit der Maus verschieben. Wenn Sie Bilder verknüpfen aktivieren, werden die Bewegungssteuerelemente (Rollbalken, Mausrad) der Bilder miteinander verknüpft.

Die Infobox im Bild zeigt Details, wie Größe, Auflösung und Farbtiefe des Bildes an. Wenn sie Ihnen im Weg liegt, kann sie mit Ansicht → Bildinformation ausgeblendet werden. Die Bildinformation wird als Hinweistext angezeigt, wenn Sie mit der Maus über den Bildtitel fahren.

Wenn die Bilder überlagert sind, wird die relative Intensität der Bilder (Alpha Überblendung) durch einen Schieber am linken Rand gesteuert. Sie können direkt auf den Schieber klicken, um einen Wert einzustellen, Sie können den Schieber auf einen bestimmten Wert ziehen oder per **Strg+Umsch**-Mausrad einstellen.

Die Schaltfläche oberhalb des Schiebers schaltet zwischen 0% und 100% Überblendung um. Wenn Sie die Schaltfläche doppelklicken, wird die Überblendung im Sekundentakt umgeschaltet, bis Sie die Schaltfläche erneut betätigen. Diese Funktion kann ihnen helfen, kleine Änderungen in Bildern zu entdecken.

Manchmal möchten Sie lieber die Unterschiede anstatt einer Überblendung angezeigt bekommen, z.B. weil Sie das Layout zweier Versionen einer Platine miteinander vergleichen wollen. Wenn Sie

die Überblendung deaktivieren, werden stattdessen die Unterschiede durch ein *XOR* der Bildpunkte hervorgehoben. Änderungen werden dadurch farbig, unveränderte Bereiche Weiß dargestellt.

4.10.5. Externe Programme

Wenn TortoiseMerge nicht all das kann, was Sie benötigen, versuchen Sie es doch mit einem der vielen Open-Source und kommerziellen Programme. Jeder hat seine Lieblingsanwendung und diese Liste ist keinesfalls vollständig, aber hier sind ein paar Beispiele, die Sie sich anschauen sollten:

WinMerge

WinMerge [<http://winmerge.sourceforge.net/>] ist ein großartiger Open Source Vergleichs- und Konflikteditor, der auch mit Verzeichnissen umgehen kann.

Perforce Merge

Perforce ist ein kommerzielles Versionskontrollsystem, aber Sie können den Vergleichs- und Konflikteditor kostenlos nutzen. Weitere Informationen finden sich auf der *Perforce* [<http://www.perforce.com/perforce/products/merge.html>] Webseite.

KDiff3

KDiff3 ist ein freier Vergleichs- und Konflikteditor, der auch Verzeichnisse vergleichen kann. Den Download finden Sie *hier* [<http://kdiff3.sf.net/>].

ExamDiff

ExamDiff Standard ist Freeware. Es kann Dateien, aber keine Verzeichnisse vergleichen. ExamDiff Pro ist Shareware und hat eine Menge an Extras einschließlich dem Vergleichen von Verzeichnissen und der Möglichkeit des Editierens. In beiden Ausführungen ab Version 3.2 ist auch Unterstützung für Unicode vorhanden. Sie können sie von *PrestoSoft* [<http://www.prestosoft.com/>] herunterladen.

Beyond Compare

Dies ist ähnlich wie ExamDiff Pro eine exzellente Shareware, die auch Verzeichnisse vergleichen kann und Unicode-Unterstützung besitzt. Sie können sie von *Scooter Software* [<http://www.scootersoftware.com/>] herunterladen.

Araxis Merge

Araxis Merge ist ein sehr nützliches kommerzielles Programm, das sowohl Vergleichen als auch Zusammenführen von Dateien und Ordnern unterstützt. Es kann einen Drei-Wege Vergleich beim Zusammenführen und besitzt Abgleichsfunktionen falls Sie die Reihenfolge von Programmblöcken vertauscht haben. Sie können es von *Araxis* [<http://www.araxis.com/merge/index.html>] herunterladen.

SciTE

Dieser Texteditor bietet farbige Syntaxhervorhebung für Standard-Diffs, was diese leichter lesbar macht. Sie können ihn *hier* [<http://www.scintilla.org/SciTEDownload.html>] herunterladen.

Notepad2

Notepad2 ist dazu gedacht, das normale Windows Notepad zu ersetzen und basiert auf dem Open Source Scintilla Edit Control. Außer zum Betrachten von Standard-Diffs ist es viel besser als das normale Notepad. Sie können es *hier* [<http://www.flos-freeware.ch/notepad2.html>] herunterladen.

Lesen Sie **Abschnitt 4.30.5, „Einstellungen für externe Programme“** für Informationen wie Sie diese Programme in TortoiseSVN einbinden/einrichten können.

4.11. Neue Dateien und Ordner hinzufügen

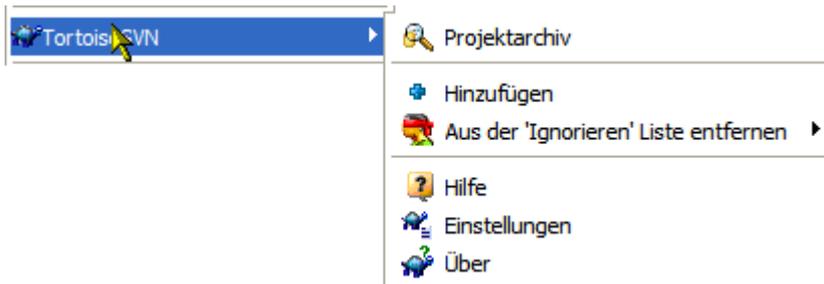


Abbildung 4.26. Explorer Kontextmenü für nicht versionierte Dateien

Wenn Sie neue Dateien und/oder Ordner während Ihrer Arbeit erstellen, müssen Sie diese auch zur Versionskontrolle hinzufügen. Wählen Sie diese Dateien und/oder Ordner aus und wählen Sie dann den Befehl TortoiseSVN → Hinzufügen... im Kontextmenü.

Nachdem Sie die Dateien/Ordner zur Versionskontrolle hinzugefügt haben, erscheinen diese mit einem Pluszeichen als überlagertes Symbol im Explorer. Dies bedeutet, dass diese bei der nächsten Übertragung mit in das Projektarchiv aufgenommen werden. Beachten Sie, dass das Hinzufügen alleine die Dateien/Ordner *nicht* im Projektarchiv speichert.



Mehrfaches Hinzufügen

Sie können den Hinzufügen-Befehl auch auf bereits versionierte Ordner anwenden. In diesem Falle zeigt Ihnen der Hinzufügen-Dialog alle noch nicht versionierten Dateien/Ordner innerhalb dieses Ordners an. Dies ist hilfreich, wenn Sie viele neue Dateien auf einmal in die Versionskontrolle aufnehmen wollen.

Wenn Sie Dateien von außerhalb Ihrer Arbeitskopie zu Ihrer Arbeitskopie hinzufügen möchten, können Sie auch die Ziehen und Ablegen Funktion verwenden:

1. Wählen Sie die Dateien zum hinzufügen
2. Ziehen Sie die Dateien mit gedrückter rechter Maustaste an den neuen Ort.
3. Lassen Sie die rechte Maustaste los.
4. Wählen Sie Kontextmenü → SVN Dateien zu dieser Arbeitskopie hinzufügen. Die Dateien werden dann in die Arbeitskopie kopiert und zur Versionskontrolle hinzugefügt.

Sie können Dateien innerhalb einer Arbeitskopie einfach hinzufügen, indem Sie sie mit der Maus auf den Übertragen-Dialog ziehen.

Wenn Sie versehentlich eine Datei oder einen Ordner zur Versionskontrolle hinzugefügt haben, können Sie dies vor dem Übertragen mittels TortoiseSVN → Hinzufügen zurücknehmen... zurücknehmen.

4.12. Dateien oder Ordner Kopieren/Umbenennen/Verschieben

Es kommt sicher häufiger vor, dass sich einige benötigte Dateien bereits an einer anderen Stelle in Ihrem Projektarchiv befinden und Sie diese einfach an den richtigen Ort kopieren wollen. Sie könnten die Dateien einfach herüberkopieren und, wie oben beschrieben, hinzufügen. Aber dabei würde die Historie der Datei verloren gehen. Wenn Sie später einen Fehler beseitigen, können Sie diese Änderungen nur dann automatisch zusammenführen, wenn die neue Datei mit dem Original in Subversion verbunden ist.

Die einfachste Methode, um Dateien und Ordner innerhalb einer Arbeitskopie zu verschieben ist das rechts-Ziehen. Wenn Sie eine Datei oder einen Ordner mit der rechten Maustaste an einen neuen Ort ziehen, erscheint beim Loslassen der Maustaste ein Kontextmenü.

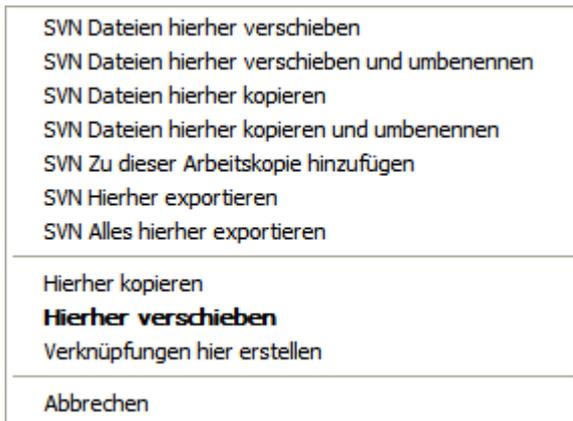


Abbildung 4.27. Rechts-Ziehen-Menü für einen Ordner unter Versionskontrolle

Damit können Sie versionierte Objekte an eine andere Stelle kopieren und dabei gegebenenfalls umbenennen.

Sie können versionierte Dateien innerhalb einer oder zwischen zwei Arbeitskopien verschieben oder kopieren. Dazu verwenden Sie das bekannte Kopieren beziehungsweise Ausschneiden in die Zwischenablage. Falls die Zwischenablage versionierte Objekte enthält, können Sie mittels TortoiseSVN → Einfügen (Achtung: Nicht das Standard Windows Einfügen), diese Objekte an einen neuen Platz in der Arbeitskopie verschieben oder kopieren.

Sie können Dateien und Ordner aus Ihrer Arbeitskopie an einen anderen Ort im Projektarchiv kopieren, indem Sie TortoiseSVN → Verzweigen/Markieren aufrufen. Lesen Sie [Abschnitt 4.19.1, „Einen Zweig oder eine Marke erstellen“](#) für weitere Details.

Sie können eine ältere Version einer Datei oder eines Ordners mit Hilfe des Log-Dialogs finden und direkt aus dem Dialog an einen neuen Ort im Projektarchiv kopieren, indem Sie Kontextmenü → Erstelle Zweig/Marke von Revision aufrufen. Lesen Sie [Abschnitt 4.9.3, „Zusätzliche Informationen erhalten“](#) für weiterführende Informationen.

Sie können auch mit Hilfe des Projektarchiv-Betrachters Objekte lokalisieren und von dort aus direkt in Ihre Arbeitskopie oder innerhalb des Projektarchivs umherkopieren. Lesen Sie in [Abschnitt 4.24, „Projektarchivbetrachter“](#) nach, wie man mit dem Betrachter arbeitet.



Kann nicht zwischen Projektarchiven kopieren

Während Sie Dateien und Ordner *innerhalb* eines Projektarchivs kopieren können, ist es *nicht möglich*, Objekte unter Beibehaltung der Historie in ein anderes Projektarchiv zu kopieren oder zu verschieben. Das geht selbst dann nicht, wenn die Projektarchive sich auf dem selben Server befinden. Alles was Sie tun können, ist den aktuellen Inhalt zu kopieren und als neuen Inhalt zum zweiten Projektarchiv hinzuzufügen.

Wenn Sie unsicher sind, ob zwei URLs auf dem gleichen Server dasselbe Projektarchiv referenzieren, rufen Sie den Projektarchivbetrachter auf und finden Sie heraus, wo die Wurzel des Projektarchivs liegt. Wenn Sie von da aus beide URLs in einem Betrachterfenster sehen können, befinden sie sich im selben Projektarchiv.

4.13. Ignorieren von Dateien und Ordnern

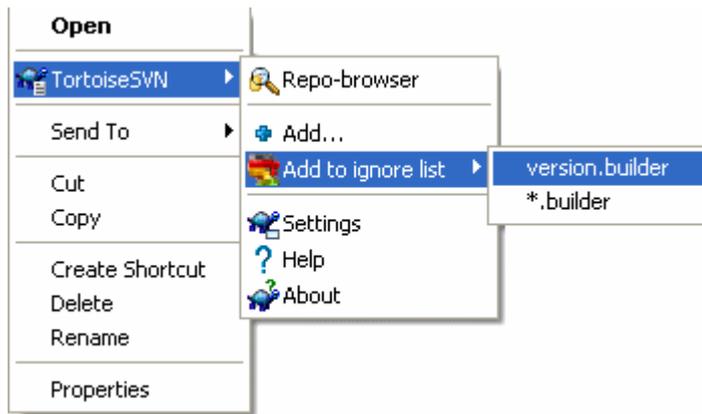


Abbildung 4.28. Explorer Kontextmenü für nicht versionierte Dateien

In den meisten Projekten werden Sie Dateien und Ordner haben, die nicht der Versionskontrolle unterstellt sein sollen. Dabei kann es sich um compilergenerierte Dateien `*.obj`, `*.lst` oder um einen Ausgabeordner für die ausführbaren Dateien handeln. Jedes Mal, wenn Sie Änderungen zum Projektarchiv übertragen, zeigt Ihnen TortoiseSVN nicht versionierte Dateien, die die Liste im Übertragen-Dialog füllen. Sie könnten natürlich die Anzeige der nicht versionierten Dateien abschalten aber dann übersehen Sie eventuell eine neue Datei, die Sie eigentlich zur Versionskontrolle hinzufügen wollten.

Die beste Möglichkeit diese Probleme zu vermeiden besteht darin, die generierten Dateien zur *Ignorieren*-Liste des Projekts hinzuzufügen. Auf diese Weise werden sie im Übertragen-Dialog niemals auftauchen, wogegen Quelldateien weiterhin angezeigt werden.

Wenn Sie mit der rechten Maustaste auf eine einzelne, nicht versionierte Datei klicken und TortoiseSVN → Ignorieren aus dem Kontextmenü wählen, wird Ihnen ein Untermenü angezeigt, das Ihnen erlaubt, nur diese Datei oder alle Dateien mit dieser Endung zu ignorieren. Wenn Sie mehrere Elemente wählen, erscheint kein Untermenü und es werden genau diese Dateien / Ordner zur *Ignorieren*-Liste hinzugefügt.

Wenn Sie einen oder mehrere Einträge aus der *Ignorieren* Liste entfernen wollen, machen Sie einen Rechtsklick und wählen Sie TortoiseSVN → Aus Ignorieren Liste entfernen. Sie können auch die `svn:ignore` Eigenschaft des Ordners direkt bearbeiten. Dies ermöglicht Ihnen, wie im folgenden Abschnitt beschrieben, mittels Platzhalterzeichen allgemeinere Definitionen vorzunehmen. Lesen Sie [Abschnitt 4.17, „Projekt-Einstellungen“](#) für weitere Information darüber, wie man Eigenschaften direkt setzt. Bitte beachten Sie, dass jeder Eintrag auf einer eigenen Zeile stehen muss. Es genügt nicht, die Einträge durch Leerzeichen zu trennen.



Die globale ignorieren Liste

Eine weitere Möglichkeit Dateien zu ignorieren ist, diese in die *globale ignorieren Liste* einzutragen. Der große Unterschied besteht darin, dass die globale Liste eine Client Eigenschaft ist. Sie gilt für *alle* Subversion Projekte, aber nur auf dem jeweiligen Anwender PC. Im allgemeinen ist es besser, die `svn:ignore` Eigenschaft zu verwenden, weil diese für einzelne Projektbereiche gesetzt werden kann und weil sie für jeden, der das Projekt auscheckt, gültig ist. Lesen Sie [Abschnitt 4.30.1, „Allgemeine Einstellungen“](#) für weitere Information.



Versionierte Objekte ignorieren

Versionierte Dateien und Ordner können niemals ignoriert werden - das ist eine Eigenschaft von Subversion. Wenn Sie versehentlich eine Datei zur Versionskontrolle hinzugefügt

haben, lesen Sie **Abschnitt B.8**, „Dateien ignorieren, die bereits unter Versionskontrolle sind“, um zu erfahren, wie Sie das rückgängig machen können.

4.13.1. Platzhalter in der Ignorieren-Liste

Die Ausschluss-Muster von Subversion verwenden Platzhalter, eine Technik die ursprünglich aus der Unix Welt stammt, um Dateien oder Ordner zu spezifizieren. Folgende Zeichen haben eine besondere Bedeutung:

*

Steht für eine beliebige (auch leere) Zeichenfolge.

?

Steht für ein beliebiges einzelnes Zeichen.

[...]

Steht für eines der Zeichen innerhalb der eckigen Klammern. Zwei durch „-“ getrennte Zeichen innerhalb der Klammern stehen für alle Zeichen innerhalb dieses Bereichs. Zum Beispiel steht [AGm-p] für A, G, m, n, o oder p.

Die Platzhaltersuche beachtet Groß-/Kleinschreibung, was unter Windows zu Problemen führen kann. Sie können auf die harte Tour erzwingen, dass Groß-/Kleinschreibung ignoriert wird, indem Sie Buchstabenpaare angeben Um z.B. *.tmp auszuschließen *. [Tt] [Mm] [Pp].

Eine offizielle Definition für Platzhalterzeichen findet sich in der IEEE für die „Shell Command Language“ im Abschnitt *Pattern Matching Notation* [http://www.opengroup.org/onlinepubs/009695399/utilities/xcu_chap02.html#tag_02_13].



Keine Pfade in der globalen ignorieren Liste

Sie sollten keine Pfadinformationen in die Muster einschließen. Der Algorithmus vergleicht mit reinen Datei- oder Ordernamen. Wenn Sie alle CVS Ordner ignorieren wollen, fügen Sie einfach CVS zur ignorieren Liste hinzu. Es ist nicht nötig, wie in älteren Versionen, CVS */CVS anzugeben. Wenn Sie alle tmp Verzeichnisse in einem prog Ordner, aber nicht in einem doc Ordner ignorieren wollen, sollten Sie die svn:ignore Eigenschaft des Ordners verwenden. Es gibt keinen zuverlässigen Weg, dies durch globale Ignoriermuster zu erreichen.

4.14. Löschen, Verschieben und Umbenennen

Im Gegensatz zu CVS erlaubt es Subversion, Dateien und Ordner umzubenennen oder zu verschieben. Deshalb gibt es im Kontextmenü Befehle für diese Operationen.

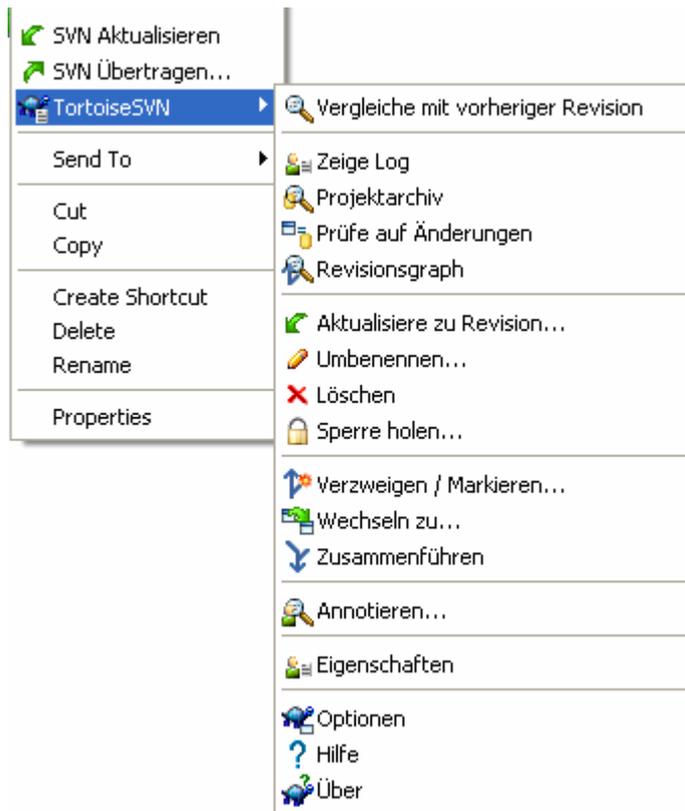


Abbildung 4.29. Explorer Kontextmenü für Dateien unter Versionskontrolle

4.14.1. Löschen von Dateien und Ordnern

Verwenden Sie TortoiseSVN → Löschen, um Dateien oder Ordner aus der Versionskontrolle zu entfernen.

Wenn Sie mittels TortoiseSVN → Löschen eine Datei löschen, wird diese sofort aus der Arbeitskopie entfernt und zum Löschen aus dem Projektarchiv bei der nächsten Übertragung markiert. Der Elternordner der Datei zeigt ein „Gelöscht“ Symbol an. Solange Sie die Löschung noch nicht übertragen haben, können Sie die Datei mit dem TortoiseSVN → Rückgängig... Befehl auf dem Elternordner wieder herstellen.

Wenn Sie mittels TortoiseSVN → Löschen einen Ordner löschen, bleibt dieser in der Arbeitskopie, wird aber durch das *Gelöscht* Symbol gekennzeichnet. Solange Sie die Löschung noch nicht übertragen haben, können Sie die Datei mit dem TortoiseSVN → Rückgängig... Befehl auf dem Ordner selbst wieder herstellen. Dieses unterschiedliche Verhalten beim Löschen von Dateien und Ordnern ist Teil von Subversion, nicht von TortoiseSVN.

Wenn Sie eine Datei oder einen Ordner aus dem Projektarchiv löschen, aber lokal noch als unversioniertes Objekt beibehalten wollen, wählen Sie Erweitertes Kontextmenü → Löschen (lokal erhalten). Sie müssen die **Umsch**-Taste gedrückt halten, während Sie einen Rechtsklick auf ein Objekt im rechten Fenster des Explorers machen, um das erweiterte Kontextmenü zu sehen.

Wenn eine *Datei* über den Explorer anstelle des TortoiseSVN Menüs gelöscht wird, zeigt Ihnen der Übertragen-Dialog die fehlenden Dateien an und erlaubt Ihnen, diese aus der Versionskontrolle zu entfernen. Wenn Sie allerdings vorher Ihre Arbeitskopie aktualisieren, wird Subversion die fehlenden Dateien bemerken und durch die aktuelle Version aus dem Projektarchiv ersetzen. Wenn Sie ein Objekt,

das sich in der Versionskontrolle befindet, löschen wollen, müssen Sie dazu TortoiseSVN → Löschen verwenden, damit Subversion nicht raten muss, was Sie wirklich tun wollten.

Wenn ein *Ordner* über den Explorer anstelle des TortoiseSVN Menüs gelöscht wird, beschädigt das Ihre Arbeitskopie und sie können diese nicht mehr übertragen. Wenn Sie Ihre Arbeitskopie aktualisieren, wird Subversion den fehlenden Ordner wieder durch die neueste Version aus dem Projektarchiv ersetzen. Sie können den Ordner dann auf die korrekte Art mittels TortoiseSVN → Löschen entfernen.



Eine gelöschte Datei/Ordner zurückholen

Wenn Sie eine Datei oder einen Ordner gelöscht und die Löschung bereits ins Projektarchiv übertragen haben, wird ein normaler TortoiseSVN → Rückgängig Befehl diese nicht wieder zurückbringen können. Aber die Datei oder der Ordner ist nicht verloren. Wenn Sie die Revisionsnummer kennen, in der sie gelöscht wurde (Benutzen Sie den Log-Dialog um das herauszufinden) öffnen Sie den Projektarchivbetrachter. Wechseln Sie dann zu der Revisionsnummer, selektieren Sie die Datei oder den gelöschten Ordner, Rechtsklick und wählen Sie den Befehl Kontextmenü → Kopieren zu.... Als Ziel für diesen Kopierbefehl geben Sie den Pfad zu Ihrer Arbeitskopie an.

4.14.2. Dateien und Ordner verschieben

Wenn Sie eine Datei oder einen Ordner an Ort und Stelle umbenennen wollen, wählen Sie Kontextmenü → Umbenennen.... Sie müssen nur noch den neuen Namen angeben.

Wenn Sie Dateien innerhalb Ihrer Arbeitskopie, zum Beispiel in einen anderen Ordner, verschieben möchten, können Sie ebenfalls die Ziehen und Ablegen Funktion verwenden:

1. Wählen Sie die Dateien aus welche Sie verschieben möchten.
2. Ziehen Sie die Dateien mit gedrückter rechter Maustaste an den neuen Ort.
3. Lassen Sie die rechte Maustaste los.
4. Wählen Sie den Befehl Kontextmenü → SVN Dateien hierher verschieben



Den übergeordneten Ordner übertragen

Da Umbenennen beziehungsweise Verschieben intern in zwei Schritten als Löschen und Hinzufügen implementiert sind, müssen Sie den übergeordneten Ordner des verschobenen/ umbenannten Objektes übertragen, damit die Löschung zum Projektarchiv übertragen wird. Wenn Sie das nicht tun, bleiben die lokal gelöschten Objekte im Projektarchiv erhalten und die Arbeitskopien Ihrer Kollegen werden beim Aktualisieren von der Löschung nichts mitbekommen. Das führt dazu, dass sie dann *beide*, die alte und die neue, Kopie des Objektes enthalten.

Sie *müssen* einen umbenannten Ordner übertragen, bevor Sie Dateien innerhalb des Ordners bearbeiten. Andernfalls kann Ihre Arbeitskopie sehr durcheinander kommen.

Sie können auch mit Hilfe des Projektarchiv-Betrachters Objekte verschieben. Lesen Sie in [Abschnitt 4.24, „Projektarchivbetrachter“](#) nach, wie man mit dem Betrachter arbeitet.



SVN Externals nicht verschieben

Sie sollten *niemals* Verzeichnisse, die auf `svn:externals` zeigen mittels TortoiseSVN Verschieben oder Umbenennen. Dies würde dazu führen, dass das Objekt aus dem

externen Projektarchiv gelöscht wird, was voraussichtlich eine Menge Leute verärgern würde. Wenn Sie einen externen Ordner verschieben wollen, sollten Sie dazu den normalen Windows Explorer verwenden und anschließend die `svn:externals` Eigenschaften des Quell- und des Zielordners anpassen.

4.14.3. Groß-/Kleinschreibung eines Dateinamens ändern

Die Groß-/Kleinschreibung eines Dateinamens mit Subversion unter Windows zu ändern ist trickreich, weil während einer kurzen Zeitspanne während des Umbenennens die Datei in beiden Schreibweisen existieren muss. Da Windows die Groß-/Kleinschreibung ignoriert, gibt es Konflikte beim Verwenden des normalen Umbenennen Befehls.

Glücklicherweise gibt es (mindestens) zwei Lösungen, eine Datei umzubenennen ohne die Historie zu verlieren. Es ist wichtig, die Datei mit Subversion umzubenennen. Ein Umbenennen nur im Explorer wird die Arbeitskopie beschädigen!

Lösung A) (empfohlen)

1. Übertragen Sie die Änderungen der Arbeitskopie.
2. Benennen Sie mit dem Projektarchivbetrachter die Datei von GROSSbuchstaben in grossBUCHSTABEN direkt im Projektarchiv um.
3. Aktualisieren Sie die Arbeitskopie.

Lösung B)

1. Benennen Sie die Datei mit dem Kontextmenü → Umbenennen... von GROSSbuchstaben in GROSSbuchstaben_ um.
2. Änderungen übertragen.
3. Benennen Sie die Datei von GROSSbuchstaben_ in grossBUCHSTABEN um.
4. Änderungen übertragen.

4.14.4. Behandeln von Konflikten in der Groß-/Kleinschreibung

Wenn das Projektarchiv bereits zwei Dateien mit demselben Namen aber unterschiedlicher Groß-/Kleinschreibung (z.B. TEST.TXT und test.txt) enthält, können Sie diesen Ordner unter Windows nicht mehr aktualisieren oder auschecken. Während Subversion bei Dateinamen Groß-/Kleinschreibung beachtet, wird sie von Windows ignoriert.

Dies passiert manchmal, wenn aus verschiedenen Arbeitskopien Dateien mit dem gleichen Namen aber unterschiedlicher Groß-/Kleinschreibung übertragen werden. Eine weitere Möglichkeit sind Übertragungen aus einem Betriebssystem wie Linux, das Groß-/Kleinschreibung berücksichtigt.

In diesem Falle müssen Sie entscheiden, welche der beiden Dateien Sie behalten möchten und die andere Datei aus dem Projektarchiv löschen oder umbenennen.



Zwei Dateien mit demselben Namen verhindern

Es gibt ein serverseitiges Aktionsskript welches verhindert, dass zwei Dateien mit demselben Namen aber unterschiedlicher Schreibweise in das Projektarchiv gelangen hier: <http://svn.collab.net/repos/svn/trunk/contrib/hook-scripts/> .

4.14.5. Externes Umbenennen reparieren

Manchmal wird Ihre freundliche Entwicklungsumgebung beim Umarbeiten von Projekten Dateien für Sie umbenennen und Subversion nicht davon in Kenntnis setzen. Wenn Sie dann Ihre Änderungen übertragen

wollen, zeigt Ihnen Subversion die alte Datei als fehlend und die neue als nicht versioniert an. Sie könnten jetzt einfach die neue Datei übertragen, aber Sie würden dadurch die Historie der Datei verlieren, da Subversion nicht weiß, dass die beiden Dateien zusammenhängen.

Besser ist es, wenn Sie Subversion mitteilen, dass diese Änderung eigentlich eine Umbenennung war. Sie können das in den Übertragen- und Auf Änderung prüfen-Dialogen tun. Markieren Sie einfach beide, die alte (fehlende) und die neue (unversionierte) Datei und wählen Sie Kontextmenü → Umbenennen reparieren, um die beiden Dateien zu einer Umbenennung zusammenzufassen.

4.14.6. Nicht versionierte Dateien löschen

Meistens tragen Sie in Ihre Ignorieren-Liste sämtliche generierten Dateien ein, da diese normalerweise von der Versionierung ausgeschlossen werden sollen. Aber was ist, wenn Sie all diese ignorierten Dateien löschen wollen, um Ihr Projekt sauber generieren zu können? Meistens gibt es eine entsprechende Funktion im Makefile, aber falls es noch kein Makefile gibt oder dieses gerade überarbeitet wird wäre es nützlich, eine Funktion zum Bereinigen der Arbeitskopie zu haben.

TortoiseSVN stellt eine solche Funktion unter Erweitertes Kontextmenü → Unversionierte Objekte löschen... zur Verfügung. Sie müssen die **Umsch** Taste drücken, während Sie einen Rechtsklick auf einen Ordner im rechten Fenster des Explorers machen, um das erweiterte Kontextmenü zu sehen. Daraufhin wird ein Dialog mit sämtlichen nicht versionierten Dateien in Ihrer Arbeitskopie angezeigt, in dem Sie die zu löschenden Objekte auswählen können.

Wenn solche Objekte gelöscht werden, werden sie in den Papierkorb verschoben, so dass Sie eine versehentlich gelöschte Datei, die eigentlich versioniert werden sollte, wieder herstellen können.

4.15. Änderungen rückgängig machen

Wenn Sie Änderungen, welche Sie an Dateien vorgenommen haben, wieder rückgängig machen wollen, wählen Sie diese Datei aus und wählen dann den Befehl TortoiseSVN → Rückgängig... aus dem Kontextmenü aus. Ein Dialog erscheint in dem alle Dateien, die geändert wurden aufgelistet sind. Wählen Sie die Dateien, die Sie in den Ursprungszustand zurückversetzen wollen und drücken Sie OK.

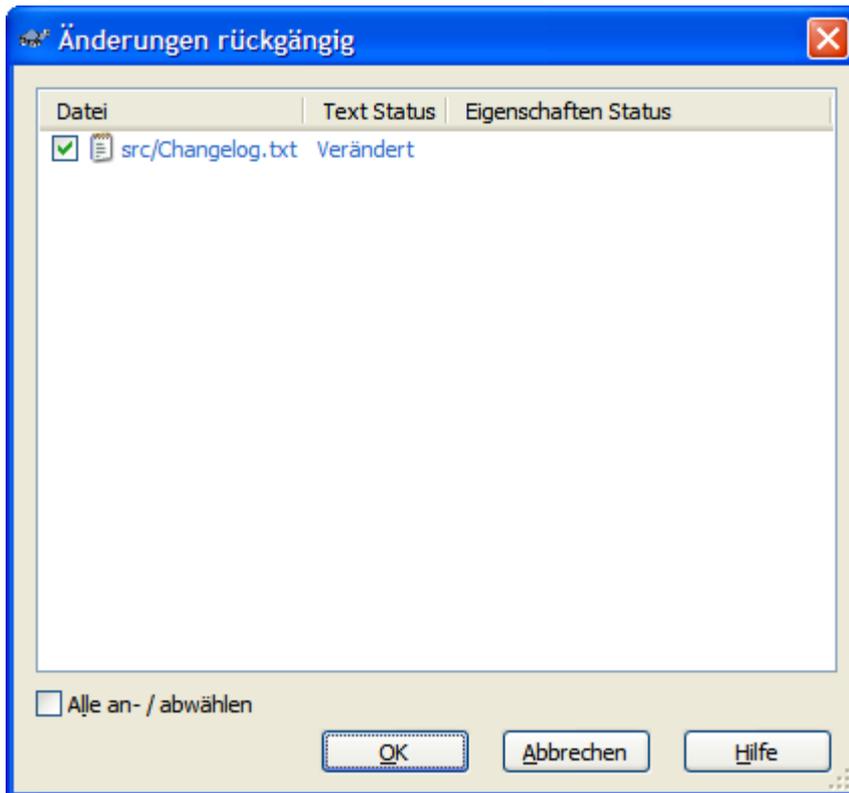


Abbildung 4.30. Rückgängig-Dialog

Wenn Sie ein umbenanntes oder gelöscht Objekt wieder herstellen möchten, müssen Sie *Rückgängig* auf dem Elternordner aufrufen, da das gelöschte Objekt für einen Rechtsklick nicht zur Verfügung steht.

Wenn Sie das Hinzufügen eines Objekts zur Versionskontrolle zurücknehmen wollen, steht dazu der Befehl TortoiseSVN → Hinzufügen zurücknehmen... zur Verfügung. Dahinter steckt in Wirklichkeit auch der Befehl Rückgängig..., aber der Name wurde der Einfachheit halber geändert.

Die Spalten in diesem Dialog können, genau wie im Auf Änderungen prüfen-Dialog, angepasst werden. Lesen Sie in [Abschnitt 4.7.3, „Prüfe auf Änderungen“](#) nach, wie das geht.



Änderungen rückgängig machen, die bereits übertragen wurden

Der Rückgängig Befehl kann nur lokale Änderungen rückgängig machen. Er kann *keine* Änderungen, welche bereits übertragen wurden rückgängig machen. Wenn Sie solche Änderungen rückgängig machen wollen, schlagen Sie dazu [Abschnitt 4.9, „Log-Dialog“](#) für weitere Informationen nach.



Rückgängig ist langsam

Wenn Sie Änderungen zurücknehmen, kommt Ihnen diese Aktion unter Umständen länger als erwartet vor. Das liegt daran, dass die geänderte Version Ihrer Datei in den Papierkorb verschoben wird, so dass Sie diese bei einem versehentlichen Zurücknehmen wieder herstellen können. Wenn allerdings der Papierkorb voll ist, benötigt Windows lange, um einen Ort für die Datei zu finden. Es gibt zwei einfache Möglichkeiten, das Problem zu lösen. Entweder sie leeren den Papierkorb oder sie deaktivieren die Option Verwende Papierkorb bei Rückgängig in den TortoiseSVN Einstellungen.

4.16. Aufräumen

Wenn ein Subversion Befehl, vielleicht durch ein Serverproblem, nicht erfolgreich abgeschlossen werden konnte, kann die Arbeitskopie in einem inkonsistenten Zustand zurückgelassen werden. In diesem Fall müssen Sie TortoiseSVN → Arbeitskopie aufräumen auf dem Ordner ausführen. Am besten tun Sie das auf dem obersten Ordner Ihrer Arbeitskopie.

Aufräumen hat einen nützlichen Nebeneffekt. Falls ein Dateidatum geändert wurde, aber der Inhalt der Datei dennoch gleich geblieben ist, kann Subversion nicht gleich feststellen ob sich die Datei geändert hat, sondern muss dann die Datei Byte für Byte mit der Basiskopie der Datei vergleichen. Wenn Sie eine Menge solcher Dateien haben, benötigt das Ermitteln des Subversion Status mehr Zeit, was viele Dialoge/ Befehle sehr verlangsamt. Ein Ausführen des Bereinigen-Befehls repariert diese „falschen“ Zeitstempel wieder und stellt die volle Geschwindigkeit der Statusprüfung wieder her.



Nutze Übertragungs-Zeitstempel

Einige frühere Versionen von Subversion hatten einen Fehler, welcher die Zeitstempel der Dateien falsch gesetzt hat, wenn Sie ein Projekt mit Dateidatum auf „Datum der letzten Übertragung“ setzen benutzt haben. Führen Sie den Aufräumen-Befehl aus, um die Arbeitskopie mit korrekten Datei-Zeitstempeln zu versehen und damit die Geschwindigkeit von Subversion-Befehlen zu erhöhen.

4.17. Projekt-Einstellungen

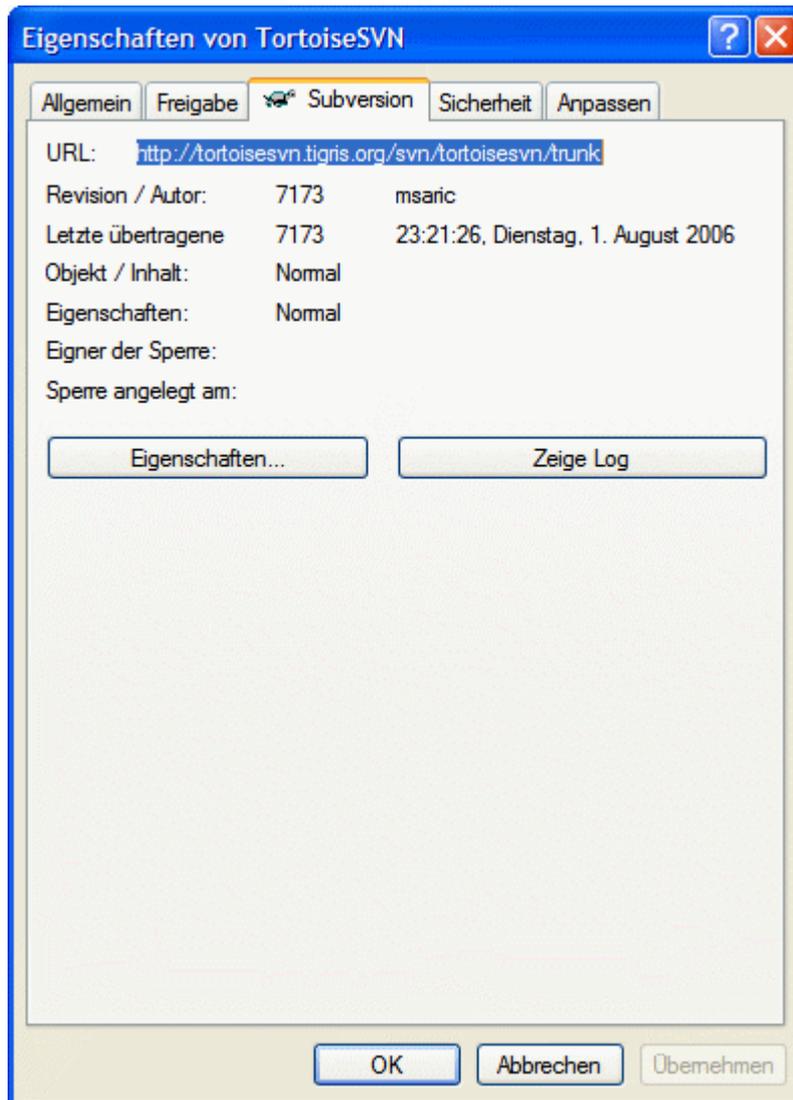


Abbildung 4.31. Explorer Eigenschaftsseite, Subversion Tab

Manchmal ist es notwendig, detailliertere Informationen über eine Datei oder einen Ordner zu haben als dies mit den überlagerten Symbolen möglich ist. Sie können diese zusätzlichen Informationen über den Eigenschaftsdialog des Explorers anzeigen. Wählen Sie dazu die Datei oder den Ordner aus, öffnen Sie das Kontextmenü mit der rechten Maustaste und wählen Sie den Eintrag **Windows Menü → Eigenschaften** aus. (Beachten Sie: dies ist der normale Eintrag im Kontextmenü des Explorers und *nicht* der Eintrag im TortoiseSVN Untermenü!). Im darauf folgenden Dialog hat TortoiseSVN einen Karteireiter für Dateien/Ordner unter Subversionkontrolle angelegt, auf dem Sie alle Informationen finden.

4.17.1. Subversion Eigenschaften

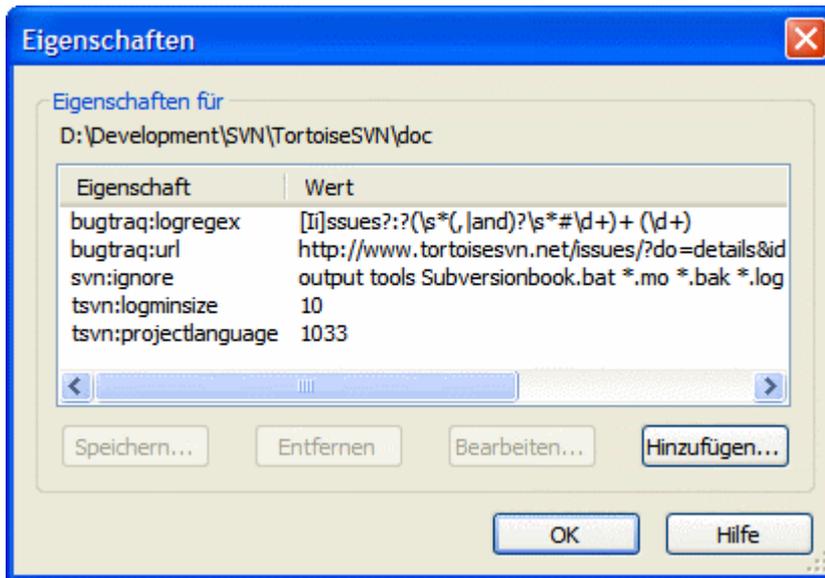


Abbildung 4.32. Subversion Eigenschaftsseite

Sie können die Subversion Eigenschaften sowohl im Windows Eigenschaftendialog als auch im Kontextmenü mit TortoiseSVN → Eigenschaften und in Statuslisten mit Kontextmenü → Eigenschaften anzeigen und verändern.

Sie können Ihre eigenen Eigenschaften oder Subversion Eigenschaften hinzufügen. Diese beginnen alle mit `svn:`. `svn:externals` ist solch eine Eigenschaft. Sehen Sie sich [Abschnitt 4.18, „Externe Objekte“](#) an, um mehr über das Thema zu erfahren.

4.17.1.1. svn:keywords

Subversion unterstützt eine CVS-ähnliche Expansion von Schlüsselwörtern, die dazu genutzt werden kann, Dateinamen und Revisionsinformationen in der Datei selbst einzubetten. Die derzeit unterstützten Schlüsselwörter sind:

`$Date$`

Datum der letzten bekannten Übertragung. Dieses Datum hängt von der Information ab, die bei der letzten Aktualisierung der Arbeitskopie erhalten wurde. Es wird *nicht* das Projektarchiv auf neuere Änderungen abgefragt.

`$Revision$`

Revision der letzten bekannten Übertragung.

`$Author$`

Autor der letzten bekannten Übertragung.

`$HeadURL$`

Die vollständige URL der Datei im Projektarchiv.

`Id`

Eine komprimierte Version der vorherigen drei Schlüsselwörter

Um mehr über diese Schlüsselwörter herauszufinden, lesen Sie den Abschnitt [svn.keywords](http://svnbook.red-bean.com/en/1.5/svn.advanced.props.special.keywords.html) [http://svnbook.red-bean.com/en/1.5/svn.advanced.props.special.keywords.html] im Subversion Buch, in dem die Schlüsselwörter und ihre Anwendung im Detail beschrieben werden.

Um mehr Informationen über Eigenschaften in Subversion zu erhalten, schauen Sie sich das entsprechende Kapitel [Special Properties](http://svnbook.red-bean.com/en/1.5/svn.advanced.props.html) [http://svnbook.red-bean.com/en/1.5/svn.advanced.props.html] im Subversion Buch an.

4.17.1.2. Eigenschaften hinzufügen und bearbeiten



Abbildung 4.33. Eigenschaften hinzufügen

Um eine Eigenschaft hinzuzufügen, klicken Sie zunächst auf **Neu...** Wählen Sie den Eigenschaftsnamen aus der Kombinationsliste oder geben Sie einen eigenen Namen ein und tragen Sie den gewünschten Wert in das darunter stehende Feld ein. Für Eigenschaften, die mehrere Werte akzeptieren z.B. `svn:ignore`, `svn:externals` geben Sie einfach mehrere Zeilen ein. Klicken Sie auf **OK**, um die Eigenschaft zur Liste hinzuzufügen.

Wenn Sie eine Eigenschaft für mehrere Objekte gleichzeitig setzen wollen, markieren Sie die Dateien / Ordner im Explorer und wählen Sie **Kontext Menü → Eigenschaften**.

Wenn Sie die Eigenschaft rekursiv für *alle* Dateien und Ordner unterhalb des aktuellen Ordners setzen wollen, wählen Sie die **Rekursiv** Option.

Manche Eigenschaften, wie z.B. `svn:needs-lock`, können nur für Dateien gesetzt werden. Deshalb sind diese Eigenschaften nicht in der Kombinationsliste für Ordner aufgeführt. Sie können solche Eigenschaften trotzdem rekursiv auf alle Dateien in einer Ordnerstruktur setzen, müssen aber den Eigenschaftsnamen selbst eingeben.

Um eine Eigenschaft zu verändern, wählen sie die Eigenschaft in der Liste der existierenden Eigenschaften aus und klicken dann auf **Bearbeiten...**

Um eine Eigenschaft zu löschen, wählen Sie die Eigenschaft zunächst aus der Liste aus. Klicken Sie anschließend auf **Entfernen**.

Die `svn:externals` Eigenschaft kann genutzt werden, um andere Projekte aus dem gleichen oder einem anderen Projektarchiv in die Arbeitskopie einzubinden. Weitere Informationen finden sich in [Abschnitt 4.18, „Externe Objekte“](#).

4.17.1.3. Eigenschaften exportieren und importieren

Häufig werden Sie wieder und wieder die gleichen Eigenschaften setzen, z.B. `bugtraq:logregex`. Um diesen Prozess zu vereinfachen und Eigenschaften von einem Projekt in ein anderes kopieren zu können, steht Ihnen die **Export/Import** Funktion zur Verfügung.

Ausgehend von der Datei bzw. dem Ordner, für die die Eigenschaften bereits gesetzt sind, wählen Sie TortoiseSVN → Eigenschaften, markieren die Eigenschaften, die Sie exportieren möchten und klicken auf Export.... Sie können anschließend eine Datei angeben, in der die Eigenschaften gespeichert werden.

Ausgehend von den Ordnern, die die Eigenschaften erhalten sollen, wählen Sie TortoiseSVN → Eigenschaften und klicken auf Import.... Über einen Dateiauswahldialog können Sie die gewünschte Datei angeben. Die importierten Eigenschaften werden nicht-rekursiv zu den Ordnern hinzugefügt.

Wenn sie Eigenschaften rekursiv hinzufügen möchten, folgen Sie den oben stehenden Schritten. Markieren Sie nacheinander die Eigenschaften im Dialog klicken Sie auf Bearbeiten... und aktivieren Sie die Eigenschaften rekursiv anwenden Option und bestätigen mit OK.

Das Importformat ist binär und spezifisch für TortoiseSVN. Sein einziger Zweck besteht darin, Eigenschaften per Import/Export zu übertragen. Deshalb besteht kein Grund, die Dateien zu bearbeiten.

4.17.1.4. Binäreigenschaften

TortoiseSVN kann binäre Eigenschaften mittels Dateien handhaben. Um einen binären Eigenschaftswert zu lesen, müssen sie diesen in eine Datei Speichern.... Um eine binäre Eigenschaft zu setzen, benutzen sie einen Hex-Editor oder ein anderes Programm, um eine Datei mit dem binären Inhalt zu erstellen, dann Laden... Sie diese aus der Datei.

Obwohl binäre Eigenschaften nicht häufig verwendet werden, sind sie doch in manchen Fällen nützlich. Sie können zum Beispiel die Vorschau einer großen Grafikkarte als binäre Eigenschaft in Subversion speichern, um sich schnell einen Eindruck des Bildes verschaffen zu können.

4.17.1.5. Eigenschaften automatisch setzen

Sie können Subversion so einrichten, dass Eigenschaften für Dateien und Ordner automatisch gesetzt werden, wenn Sie zum Projektarchiv hinzugefügt werden. Es gibt zwei verschiedene Wege dazu.

Sie können die Subversion Konfigurationsdatei bearbeiten, um diese Funktion auf dem Client zu aktivieren. Auf der Allgemein Seite der TortoiseSVN Einstellungen findet sich eine Bearbeiten Schaltfläche, die Sie direkt dort hinbringt. Die Konfigurationsdatei mit der einige Subversion Verhaltensweisen festgelegt werden ist eine einfache Textdatei. Sie müssen darin zwei Dinge ändern. Erstens müssen Sie in der miscellany Sektion die Zeile `enable-auto-props = yes` auskommentieren. Zweitens müssen Sie in dem darunterstehenden Bereich festlegen, welche Eigenschaften zu welchen Dateitypen hinzugefügt werden sollen. Dabei handelt es sich um eine Subversion Standardfunktion, die in jedem Subversion Client funktioniert. Allerdings müssen die Einstellungen auf jedem Computer vorgenommen werden. Es gibt keine Möglichkeit die Einstellungen über das Projektarchiv zu verbreiten.

Eine alternative Methode ist, die `tsvn:autoprops` Eigenschaft, wie im nächsten Abschnitt beschrieben, auf Ordner zu setzen. Diese Methode funktioniert nur mit TortoiseSVN clients, dafür wird die Eigenschaft beim Aktualisieren in alle Arbeitskopien verteilt.

Egal welche Methode Sie wählen, Sie sollten wissen, dass auto-props nur beim Hinzufügen von Dateien zum Projektarchiv gesetzt werden. Es werden niemals die Eigenschaften von Dateien gesetzt oder geändert, die bereits versioniert sind.

Wenn Sie ganz sicher sein wollen, dass neue Dateien die korrekten Eigenschaften erhalten, sollten Sie eine `pre-commit` Aktion im Projektarchiv einrichten, die Übertragungen verwirft, wenn die erforderlichen Eigenschaften nicht gesetzt sind.



Übertrage Eigenschaften

Subversion Eigenschaften sind auch unter Versionskontrolle. Nachdem Sie eine Eigenschaft geändert haben, müssen Sie die Änderungen zum Projektarchiv übertragen.



Konflikte in Eigenschaften

Sollte es beim Übertragen zum Server zu einem Konflikt kommen, weil ein anderer Benutzer die gleiche Eigenschaft geändert hat, generiert Subversion eine `.prej` Datei. Löschen Sie diese Datei, nachdem Sie den Konflikt aufgelöst haben.

4.17.2. TortoiseSVN Projekteigenschaften

TortoiseSVN besitzt ein paar eigene spezielle Eigenschaften, die mit `tsvn:` beginnen.

- `tsvn:logminsize` Setzt die minimale Länge einer Logmeldung für eine Übertragung. Wenn Sie eine Logmeldung eingeben welche kürzer ist als mit dieser Eigenschaft festgelegt, dann bleibt die Übertragung deaktiviert. Dies ist nützlich wenn Sie die Benutzer dazu anhalten möchten, für jede Übertragung eine gute Logmeldung zu schreiben. Wenn diese Eigenschaft nicht gesetzt ist, sind leere Logmeldungen erlaubt.

`tsvn:lockminsize` Setzt die minimale Länge einer Sperrmeldung. Wenn Sie eine Meldung eingeben, die kürzer ist als mit dieser Eigenschaft festgelegt, dann bleibt die Sperrung deaktiviert. Dies ist nützlich, wenn Sie die Benutzer dazu anhalten möchten, für jede Sperrung eine Beschreibung anzugeben. Wenn diese Eigenschaft nicht gesetzt ist, sind leere Sperrmeldungen erlaubt.

- `tsvn:logwidthmarker` kann für Projekte benutzt werden, welche Logmeldungen mit einer maximalen Zeilenlänge (typischerweise 80 Zeichen) haben möchten. Wenn diese Eigenschaft auf einen Wert ungleich 0 gesetzt wird, wird einerseits der Wortumbruch für die Logmeldung abgeschaltet und andererseits eine vertikale Linie gezeichnet um die maximale Zeilenlänge anzuzeigen. Auf diese Weise können Sie jederzeit sehen, ob Ihre Logmeldung die maximale Zeilenlänge schon überschreitet oder nicht. Beachten Sie, dass dies nur korrekt funktioniert, wenn Sie eine Schrift mit fester Breite für das Eingabefeld ausgewählt haben.
- `tsvn:logtemplate` wird für Projekte benutzt, die Regeln für die Formatierung der Logmeldungen haben. Diese Eigenschaft enthält einen mehrzeiligen Text, der automatisch im Übertragen-Dialog in die Textbox eingefügt wird. Sie können dann diesen Text bei der Übertragung bearbeiten. Wenn Sie gleichzeitig `tsvn:logminsize` verwenden, stellen Sie sicher dass dieser Wert größer als die Länge der Schablone ist, da sonst dieser Schutz wegfällt
- Subversion erlaubt Ihnen, `autoprops` anzugeben, die für jede neu hinzugefügte oder importierte Datei, abhängig von der Dateiendung, gesetzt werden. Das erfordert, dass jeder Client die gleichen `autoprops` in seiner Subversion Konfigurationsdatei stehen hat. `tsvn:autoprops` können auch auf Ordner gesetzt werden und werden mit den lokalen `autoprops` des Anwenders zusammengeführt, wenn Dateien zu einem Projektarchiv hinzugefügt oder importiert werden. Das Format ist dasselbe wie bei den Subversion `autoprops`, z.B. setzt `*.sh = svn:eol-style=native;svn:executable` zwei Eigenschaften auf Dateien mit der `.sh` Endung.

Wenn es einen Konflikt zwischen den lokalen `autoprops` und `tsvn:autoprops` gibt, haben die Projekteinstellungen Vorrang vor den allgemeinen Einstellungen.

- Im Übertragen-Dialog haben Sie die Möglichkeit, die Liste der geänderten Dateien inklusive des Status (hinzugefügt, geändert, etc) einzufügen. `tsvn:logfilelistenglish` legt fest, ob die vordefinierten Logmeldungen in englisch oder in einer anderen Sprache eingefügt werden sollen. Der Standardwert ist `true`.
- TortoiseSVN kann zur Rechtschreibprüfung die Wörterbücher benutzen, welche auch von OpenOffice oder Mozilla verwendet werden. Wenn Sie ein solches installiert haben, legt diese Eigenschaft fest, welches Sprachmodul für das Projekt benutzt werden soll. `tsvn:projectlanguage` setzt die Sprache des zu benutzenden Wörterbuchs. Die Werte für die Sprache können Sie hier nachschlagen: [MSDN: Language Identifiers](http://msdn2.microsoft.com/en-us/library/ms776260.aspx) [http://msdn2.microsoft.com/en-us/library/ms776260.aspx].

Sie können diesen Wert als Dezimalzahl oder, wenn Sie das bevorzugen, als Hexadezimalzahl mit dem Prefix 0x angeben. Zum Beispiel Englisch (US) kann als 0x0409 oder 1033 eingegeben werden.

- Die Eigenschaft `tsvn:logsummary` wird verwendet, um einen Abschnitt aus der Logmeldung zu extrahieren, der dann als Zusammenfassung der Logmeldung im Log-Dialog angezeigt wird.

Die `tsvn:logsummary` Eigenschaft muss einen einzeiligen regulären Ausdruck enthalten, der eine RegEx-Gruppe enthält. Der Inhalt dieser Gruppe wird als Zusammenfassung verwendet

Ein Beispiel: `\[ZUSAMMENFASSUNG\]:\s+(.*)` Dieser Ausdruck wird alles nach „[ZUSAMMENFASSUNG]“ aus der Logmeldung ausschneiden und als Zusammenfassung verwenden.

- Wenn Sie eine neue Eigenschaft hinzufügen wollen, können Sie entweder eine Standardeigenschaft aus der Kombinationsliste wählen oder einen beliebigen Eigenschaftsnamen eingeben. Falls für Ihr Projekt spezifische Eigenschaften verwendet werden, die ebenfalls in der Auswahlliste erscheinen sollen, können Sie mit den TortoiseSVN Eigenschaften `tsvn:userfileproperties` und `tsvn:userdirproperties` eine Liste von Projekteigenschaften vorgeben. Wenden Sie die TortoiseSVN Eigenschaften auf einen Ordner an. Sobald Sie anschließend die Eigenschaften eines Unterordners bearbeiten, werden die Projekteigenschaften in der Liste der vorgegebenen Eigenschaftsnamen erscheinen.

Manche `tsvn:` Eigenschaften erfordern einen `true/false` Wert. TortoiseSVN versteht auch `yes` als Synonym für `true` und `no` als Synonym für `false`.

TortoiseSVN kann einige Fehlerverfolgungssysteme in Subversion integrieren. Dies geschieht über Eigenschaften, die mit `bugtraq:` beginnen. Wie Sie mit diesen Eigenschaften umgehen ist in [Abschnitt 4.28, „Integration mit einem System zur Fehlerverfolgung“](#) erklärt.

TortoiseSVN kann auch mit einigen webbasierten Projektarchivbetrachtern, unter Verwendung von Eigenschaften, die mit `webviewer:` beginnen, zusammenarbeiten. Lesen Sie [Abschnitt 4.29, „Integration mit webbasierten Projektarchivbetrachtern“](#) für weitere Informationen.



Setzen Sie die Projekteigenschaften auf Ordner

Diese speziellen Eigenschaften müssen für Ordner und nicht für Dateien gesetzt werden. Wenn Sie eine Datei oder einen Ordner übertragen dann werden die Eigenschaften aus eben diesem Ordner gelesen. Wenn die Eigenschaften nicht gefunden werden, sucht TortoiseSVN im Ordnerbaum aufwärts nach diesen Eigenschaften, bis entweder die Eigenschaften gefunden wurden oder der oberste Ordner der Arbeitskopie gefunden wurde. Wenn Sie also sicher sind dass alle Benutzer ihre Arbeitskopie z.B. von `/trunk` ausgecheckt haben, reicht es aus die Eigenschaften dort zu setzen. Wenn Sie hingegen nicht sicher sein können, dann müssen Sie diese Eigenschaften rekursiv auf alle Unterordner Ihres Projektes setzen. Beachten Sie: eine Eigenschaft welche auf Unterordnern gesetzt ist überschreibt dieselbe Eigenschaft für übergeordnete Ordner.

Ausschließlich für Projekteigenschaften können Sie die *Rekursiv* Option wählen, um die Eigenschaft auf alle Unterordner zu übertragen, ohne sie gleichzeitig auch für alle Dateien zu setzen.

Wenn Sie mit TortoiseSVN neue Unterordner hinzufügen, werden sämtliche, auf den Elternordner gesetzte, Projekteigenschaften zum Unterordner hinzugefügt.



Achtung

Obwohl die TortoiseSVN Projekteigenschaften sehr nützlich sind, gelten sie nur für TortoiseSVN und einige von ihnen sogar nur für neuere Versionen von TortoiseSVN.

Sobald Projektmitarbeiter verschiedene Subversion Clients oder eventuell alte Versionen von TortoiseSVN einsetzen, sollten Sie besser mit Aktionsskripten im Projektarchiv Projektregeln durchsetzen. Projekteigenschaften können nur beim Einrichten von Regeln helfen, jedoch nicht dabei, diese durchzusetzen.

4.18. Externe Objekte

Manchmal ist es nützlich eine Arbeitskopie zu haben, die aus mehreren Projekten besteht. Zum Beispiel kann es vorkommen, dass Sie Unterordner haben wollen, die von verschiedenen anderen Stellen des Projektarchivs kommen oder vielleicht sogar aus verschiedenen Projektarchiven. Wenn Sie wollen, dass jeder Benutzer die gleiche Struktur der Arbeitskopie hat, können Sie mit Hilfe der `svn:externals` Eigenschaft die gewünschten Ressourcen an den benötigten Stellen in der Arbeitskopie einbinden.

4.18.1. Externe Ordner

Nehmen wir an Sie erstellen eine Arbeitskopie von `/projekt1` in `D:\dev\projekt1`. Selektieren Sie den Ordner `D:\dev\projekt1`, machen Sie einen Rechtsklick und wählen Sie **Windows Menü** → **Eigenschaften** aus dem Kontextmenü. Der Eigenschaften-Dialog erscheint, auf dessen Subversion Tab Sie Eigenschaften anschauen, verändern oder setzen können. Klicken Sie auf **Hinzufügen...** und wählen Sie die `svn:externals` Eigenschaft aus der Liste und schreiben Sie die zu referenzierende Projektarchiv URL in dem Format `Name URL` oder, wenn Sie eine bestimmte Revision angeben wollen, `Name -rREV URL` in das Textfeld. Sie können mehrere externe Projekte - eines pro Zeile - hinzufügen. Nehmen wir an, Sie haben folgende Eigenschaften auf `D:\dev\projekt1` gesetzt:

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

Klicken Sie auf **Setzen** und übertragen Sie Ihre Änderungen. Sobald jetzt jemand seine Arbeitskopie aktualisiert, wird Subversion einen Unterordner `D:\dev\projekt1\sounds` anlegen und in diesen das „Sounds“ Projekt auschecken, einen weiteren Unterordner `D:\dev\projekt1\quick_graphs` mit dem „Graphics“ Projekt, und schließlich einen verschachtelten Unterordner `D:\dev\projekt1\skins\toolkit`, der Revision 21 des „Skinmaker“ Projekts enthält.

Sonderzeichen in URLs müssen korrekt ersetzt werden Leerzeichen z.B. durch `%20`, wie im obigen Beispiel, damit die externen Verweise funktionieren.

Wenn der lokale Pfad Leerzeichen oder andere Sonderzeichen enthalten soll, können Sie ihn in doppelte Anführungszeichen setzen oder, im Unix Stil, einen `\` (umgekehrter Schrägstrich) jedem Sonderzeichen voranstellen. Dies bedeutet auch, dass Sie den normalen Schrägstrich `/` als Pfadtrennzeichen verwenden müssen. Beachten Sie, dass dieses Verhalten mit Subversion 1.6 eingeführt wurde und mit älteren Clients nicht funktioniert.



Nutze explizite Revisionsnummern

Sie sollten in Erwägung ziehen, in allen „externals“ Definitionen explizite Revisionsnummern anzugeben. Diese Vorgehensweise ermöglicht es Ihnen, exakt zu bestimmen zu welchem Zeitpunkt welcher Stand der externen Informationen herangezogen wird. Neben dem sofort zu erkennenden Aspekt, dass Sie nicht mehr von Änderungen in externen Bibliotheken, über die Sie keine Kontrolle haben, überrascht werden können, bietet die Verwendung expliziter Revisionsnummern den Vorteil, dass, wenn Sie Ihre Arbeitskopie zurückdatieren, auch die externen Verweise entsprechend zurückdatiert werden. Andernfalls enthalten die externen Verweise die HEAD Revision. Für ein Softwareprojekt kann das den Unterschied zwischen einem erfolgreichen und einem fehlgeschlagenen Erzeugen eines älteren Projektstandes ausmachen.



Ältere svn:externals Definitionen

Das hier angezeigte Format wurde in Subversion 1.5 eingeführt. Sie können auch auf das ältere Format stoßen, das dieselbe Information in einer anderen Reihenfolge enthält. Das neue Format wird bevorzugt, da es einige nützliche Möglichkeiten bietet, die unten beschrieben werden. Allerdings funktioniert das neue Format nicht mit älteren Subversion Clients. Die Unterschiede werden im *Subversion Buch* [<http://svnbook.red-bean.com/en/1.5/svn.advanced.externals.html>] beschrieben.

Wenn das externe Projekt im selben Projektarchiv ist, werden alle Änderungen in diesem externen Projekt im Übertragen-Dialog aufgelistet und gemeinsam mit dem Hauptprojekt übertragen.

Wenn ein externes Projekt in einem anderen Projektarchiv ist, werden Sie zwar über Änderungen im externen Projekt informiert, Sie müssen diese jedoch separat übertragen.

Wenn Sie absolute URLs in Ihren `svn:externals` Verweisen verwenden und Sie Ihre Arbeitskopie umplatzen, (weil sich z.B. die URL des Projektarchivs ändert), ändern sich die Verweise und funktionieren unter Umständen nicht mehr.

Um solche Probleme zu vermeiden, unterstützen Subversion Clients ab Version 1.5 relative externe URLs. Es stehen vier verschiedene Methoden zur Verfügung, um externe URLs festzulegen. In den folgenden Beispielen nehmen wir zwei Projektarchive an: Eines in `http://example.com/svn/repos-1` und ein weiteres in `http://example.com/svn/repos-2`. Wir haben `http://example.com/svn/repos-1/projekt/trunk` in `C:\Projekt` ausgecheckt und die `svn:externals` Eigenschaft ist auf `trunk` gesetzt.

Relativ zum Elternverzeichnis

Diese URLs beginnen stets mit der Zeichenkette `../` zum Beispiel:

```
../grafik/foo gemeinsam/foo-grafik
```

Dadurch wird `http://example.com/svn/repos-1/grafik/foo` in `C:\Projekt\gemeinsam\foo-grafik` extrahiert.

Beachten Sie, dass die URL relativ zu der URL des Verzeichnisses mit der `svn:externals` Eigenschaft ist und nicht zu dem Verzeichnis in dem der externe Verweis auf die Festplatte gespeichert wird.

Relativ zur Wurzel des Projektarchivs

Diese URLs beginnen stets mit der Zeichenkette `^/` zum Beispiel:

```
^/grafik/foo gemeinsam/foo-grafik
```

Dadurch wird `http://example.com/svn/repos-1/grafik/foo` in `C:\Projekt\gemeinsam\foo-grafik` extrahiert.

Sie können einfach auf andere Projekte innerhalb desselben `SVNParentPath` (Ein gemeinsames Elternverzeichnis, das mehrere Projektarchive enthält) verweisen. Zum Beispiel:

```
^/../repos-2/werkzeug/hammer gemeinsam/hammer-werkzeug
```

Dadurch wird `http://example.com/svn/repos-2/werkzeug/hammer` in `C:\Projekt\gemeinsam\hammer-werkzeug` extrahiert.

Relativ zum Schema

URLs die mit der Zeichenkette `//` beginnen, kopieren nur den Schemateil der URL. Das ist dann nützlich, wenn auf denselben Rechnernamen, abhängig vom Netzwerk mit einem anderen Schema zugegriffen werden muss. So können zum Beispiel Clients im Intranet per `http://` zugreifen, während externe Clients `svn+ssh://` verwenden müssen. Zum Beispiel:

```
//example.com/svn/repos-1/grafik/foo gemeinsam/foo-grafik
```

Dadurch wird in Abhängigkeit vom Schema, das verwendet wurde, um `C:\Projekt` auszuchecken, `http://example.com/svn/repos-1/grafik/foo` oder `svn+ssh://example.com/svn/repos-1/grafik/foo` extrahiert.

Relativ zum Rechnernamen des Servers

URLs die mit der Zeichenkette `/` anfangen, kopieren das Schema und den Rechnernamen der URL, zum Beispiel:

```
/svn/repos-1/grafik/foo gemeinsam/foo-grafik
```

Dadurch wird `http://example.com/svn/repos-1/grafik/foo` in `C:\Projekt\gemeinsam\foo-grafik` extrahiert. Wenn Sie jedoch Ihre Arbeitskopie von einem anderen Server in `svn+ssh://spiegel.server.net/svn/repos-1/projekt1/trunk` auschecken, wird der externe Verweis von `svn+ssh://spiegel.server.net/svn/repos-1/grafik/foo` geholt.

Sie können auch, falls erforderlich, eine Stützrevision nach der URL angeben, z.B. `http://sounds.red-bean.com/repos@19`.

Wenn sie mehr Informationen brauchen, wie TortoiseSVN Eigenschaften behandelt, lesen Sie [Abschnitt 4.17, „Projekt-Einstellungen“](#).

Um sich über verschiedene Methoden zum Zugriff auf gemeinsame Unterprojekte zu informieren, lesen Sie [Abschnitt B.6, „Ein gemeinsames Unterprojekt einbinden“](#).

4.18.2. Externe Dateien

Beginnend mit Subversion 1.6 können Sie einzelne externe Dateien unter Verwendung der gleichen Syntax wie für Ordner an Ihre Arbeitskopie binden. Es gibt jedoch ein paar Einschränkungen.

- Beim Verweis auf eine externe Datei muss dafür Sorge getragen werden, dass diese in einem bereits existierenden, versionierten Ordner zu liegen kommt. Im allgemeinen ist es am sinnvollsten, die Datei direkt in dem Ordner abzulegen der den `svn:externals` Verweis enthält, es kann aber auch ein versionierter Unterordner sein. Im Gegensatz dazu wird bei Verweisen auf externe Verzeichnisse, falls erforderlich, die Hierarchie aus nicht versionierten Ordnern automatisch aufgebaut.
- Die URL für die externe Datei muss sich in demselben Projektarchiv befinden, wie die URL in die die externe Datei eingebettet wird. Verweise auf externe Dateien zwischen Projektarchiven werden nicht unterstützt.

Eine externe Datei verhält sich in den meisten Fällen wie jede andere versionierte Datei, sie kann aber nicht mit den normalen Befehlen gelöscht oder verschoben werden. Stattdessen muss die `svn:externals` Eigenschaft verändert werden.



Die Unterstützung von externen Dateien ist in Subversion 1.6 unvollständig

In Subversion 1.6 ist es nicht möglich, eine externe Datei aus Ihrer Arbeitskopie zu entfernen, nachdem Sie sie hinzugefügt haben, selbst wenn Sie die `svn:externals` Eigenschaft vollständig löschen. Sie müssen eine neue Arbeitskopie auschecken, um die Datei loszuwerden.

4.19. Verzweigen / Markieren

Eine der Funktionen von Versionskontrollsystemen ist die Möglichkeit bestimmte isolierte Änderungen auf einer separaten Entwicklungslinie zu machen. Solche Linien werden *Zweige* genannt. Zweige werden üblicherweise dazu benutzt, um neue Funktionen auszutesten, ohne dadurch die normale Entwicklung am Projekt zu stören. Sobald dann eine solche neue Funktion stabil genug ist, wird der Zweig mit dem Stamm (`trunk`) zusammengeführt.

Eine andere wichtige Funktion ist die Möglichkeit, bestimmte Zustände zu markieren (z.B. eine Release-Version), damit es jederzeit möglich ist diesen bestimmten Zustand wieder herstellen zu können. Dieser Vorgang wird markieren genannt.

Subversion kennt keinen speziellen Befehl für das Verzweigen oder Markieren. Subversion benutzt stattdessen so genannte „billige Kopien“. Diese funktionieren ähnlich wie Verweise, d.h. es wird keine richtige Kopie erstellt sondern nur eine Verweis auf eine bestimmte Revision. Daraus ergibt sich, dass Zweige oder Marken schnell zu erstellen sind und nahezu keinen zusätzlichen Platz im Projektarchiv benötigen.

4.19.1. Einen Zweig oder eine Marke erstellen

Wenn Sie beim Importieren Ihres Projektes die empfohlene Ordnerstruktur verwendet haben, ist ein Verzweigen oder Markieren sehr einfach:

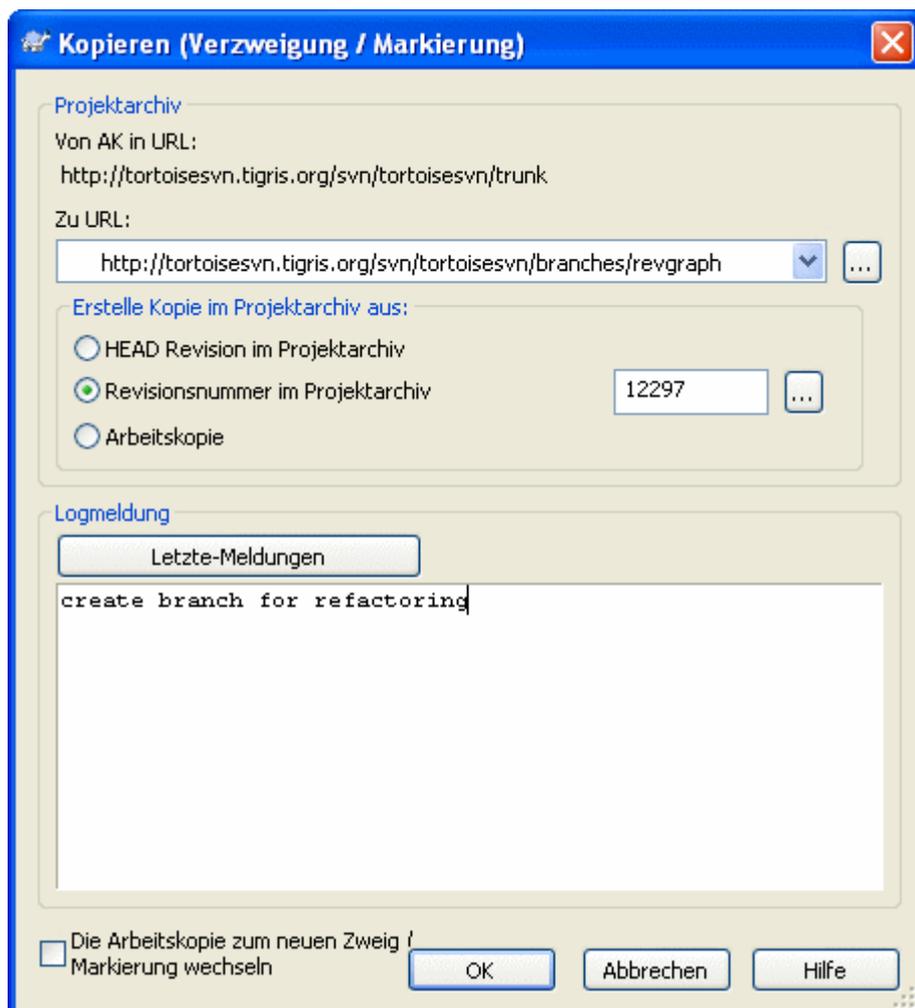


Abbildung 4.34. Der Verzweigen/Markieren-Dialog

Wählen Sie den Ordner Ihrer Arbeitskopie von dem Sie einen Zweig oder eine Marke erstellen wollen und wählen Sie den Befehl TortoiseSVN → Verzweigen/Markieren... im Kontextmenü.

Standardmäßig wird die Ziel-URL mit der Quell-URL Ihrer Arbeitskopie vorbelegt. Sie müssen nun diese URL in den Pfad Ihrer neuen Verzweigung oder Markierung ändern. Anstelle von

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

könnte das neue Verzeichnis wie folgt lauten:

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

Wenn Sie sich nicht mehr daran erinnern, welchen Namen Sie zuletzt vergeben hatten, klicken Sie einfach auf die rechte Schaltfläche. Damit öffnet sich der Projektarchivbetrachter und Sie können sich die Struktur des Projektarchivs ansehen.

Nun haben Sie mehrere Möglichkeiten, die Quelle der Kopie auszuwählen:

HEAD Revision im Projektarchiv

Der neue Zweig wird direkt im Projektarchiv als Kopie der HEAD Revision erstellt. Dabei werden keine Daten von Ihrer Arbeitskopie in das Projektarchiv übertragen und der Zweig kann sehr schnell erzeugt werden.

Revisionsnummer im Projektarchiv

Der neue Zweig wird direkt im Projektarchiv kopiert, Sie können aber auch eine ältere Revision auswählen. Das ist dann nützlich, wenn Sie z.B. vergessen haben eine Marke zu erstellen, als das Release in der letzten Woche erstellt wurde. Wenn Sie sich nicht mehr an die Revisionsnummer erinnern können, dann klicken Sie auf die Schaltfläche rechts und wählen die Revisionsnummer aus. Auch dabei werden keine Daten aus Ihrer Arbeitskopie an das Projektarchiv übertragen, so dass der Zweig sehr schnell erstellt werden kann.

Arbeitskopie

Der neue Zweig ist eine identische Kopie Ihrer lokalen Arbeitskopie. Egal ob Sie einige Dateien durch die einer älteren Revision ersetzt haben, oder sie geändert haben, diese Dateien sind genau die, die in diesen Zweig kopiert werden. Natürlich erfordert diese aufwändige Markierung, dass Daten von Ihrer Arbeitskopie zurück in das Projektarchiv übertragen werden müssen, sofern sie dort noch nicht existieren.

Wenn Sie wollen, dass Ihre Arbeitskopie sofort zum neuen Zweig/Marke wechselt, aktivieren Sie die Option **Arbeitskopie zum neuen Zweig/Marke wechseln**. Wenn Sie das tun, stellen Sie bitte vorher sicher, dass Ihre Arbeitskopie keine Änderungen enthält. Wenn sie Änderungen enthält, werden diese beim Wechseln mit dem Zweig zusammengeführt.

Klicken Sie **OK**, um die neue Kopie in das Projektarchiv zu übertragen und vergessen Sie dabei nicht, eine Logmeldung anzugeben. Beachten Sie, dass diese Kopie *innerhalb des Projektarchivs* angelegt wird.

Beachten Sie, dass das Anlegen eines Zweiges oder einer Marke sich *nicht* auf Ihre Arbeitskopie auswirkt, es sei denn Sie haben die Option zum Wechseln auf den Zweig gleich mit aktiviert. Auch wenn Sie den Zweig aus der Arbeitskopie heraus erstellen, werden die Änderungen darin in den neuen Zweig übertragen und nicht in den Stamm. Die geänderten Dateien Ihrer Arbeitskopie bleiben als gegenüber dem Stamm verändert markiert.

4.19.2. Auschecken oder Wechseln...

...das ist hier (eigentlich nicht) die Frage. Während beim Auschecken eine neue Arbeitskopie (z.B. des neu erstellten Zweiges) erstellt wird, verändert der Befehl **TortoiseSVN → Wechseln zu...** Ihre bestehende Arbeitskopie so, dass diese genau wie der erstellte Zweig aussieht. Dabei werden nur die Unterschiede zwischen Ihrer bestehenden Arbeitskopie und dem Zweig des Projektarchivs übertragen, was sich natürlich günstig auf die Netzauslastung und Ihre Geduld auswirkt.

Es gibt mehrere Möglichkeiten, mit Ihrer neuen Verzweigung oder Markierung weiterzuarbeiten. Sie können:

- **TortoiseSVN → Auschecken**, um eine neue Arbeitskopie in einem leeren Verzeichnis zu erstellen. Sie können von einem Projektarchiv so viele Arbeitskopien erstellen, wie Sie möchten.
- **Wechseln Sie mit Ihrer Arbeitskopie zu Ihrer neuen Kopie im Projektarchiv**. Dazu wählen Sie **TortoiseSVN → Wechseln zu...** aus dem Kontextmenü.

Im folgenden Dialog geben Sie die URL des Zweiges an, den Sie gerade erzeugt haben. Wählen Sie die Option **HEAD Revision** und bestätigen Sie mit **OK**. Ihre Arbeitskopie wird nun auf den neuen Zweig / die neue Marke umgestellt.

„Wechseln zu“ wird genau so wie „Aktualisieren“ niemals lokale Änderungen verwerfen. Sämtliche noch nicht übertragene Änderungen werden beim Wechseln mit den Daten aus dem Projektarchiv zusammengeführt. Wenn Sie das nicht wünschen, müssen Sie entweder die Änderungen übertragen oder verwerfen, bevor Sie „Wechseln zu“ aufrufen.

- Wenn Sie auf dem Stamm und einem Zweig gleichzeitig arbeiten wollen, aber nicht auf ein vollständiges Auschecken warten wollen, können Sie mit dem Explorer Ihr Arbeitskopie kopieren und mit der Kopie über TortoiseSVN → Wechseln zu... auf den Zweig wechseln.

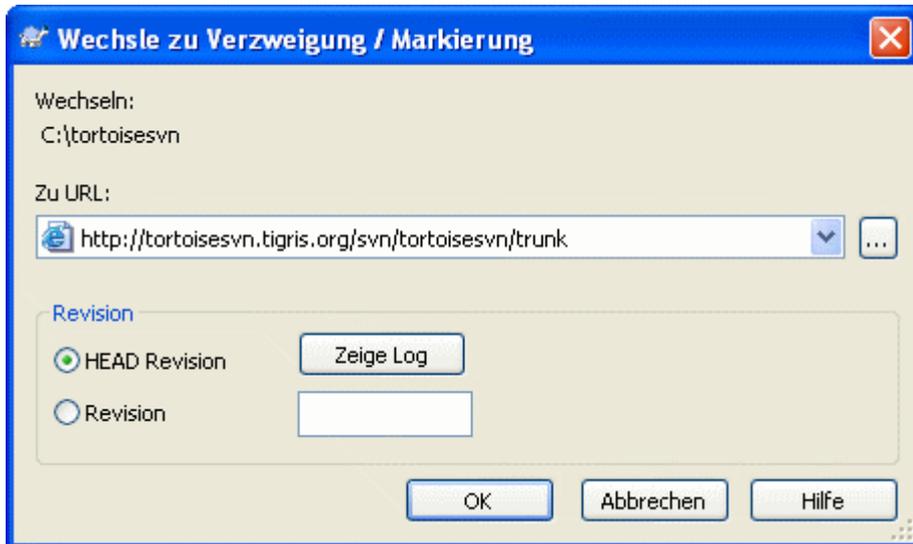


Abbildung 4.35. Der Wechseln-Zu-Dialog

Obwohl Subversion keine Unterscheidung zwischen tags und branches macht, werden sie normalerweise unterschiedlich genutzt.

- Tags (=Marken) werden typischerweise dafür verwendet, einen Entwicklungszeitpunkt einzufrieren. Als solche werden sie normalerweise nicht für die Weiterentwicklung genutzt. Dafür sind branches gedacht. Das ist auch der Hauptgrund, warum wir die /trunk /branches /tags Struktur für Projektarchive empfehlen. Änderungen an einer markierten Revision vorzunehmen ist *keine gute Idee*, aber da die Daten in Ihrer Arbeitskopie nicht schreibgeschützt sind, kann es aus Versehen doch passieren. TortoiseSVN wird Sie jedoch warnen, wenn Sie versuchen, Änderungen in einem Pfad im Projektarchiv zu übertragen, den die Zeichenfolge /tags enthält.
- Möglicherweise wollen Sie doch weitere Änderungen an einer Revision vornehmen, die Sie bereits markiert haben. Die korrekte Vorgehensweise ist, zunächst per Verzweigen/Markieren einen neuen Zweig aus der Marke (z.B. /branches/Version_1.0.x aus /tags/Version_1.0) zu erzeugen und die Änderungen auf dem Zweig durchzuführen. Wenn Sie Ihre Änderungen durchgeführt haben, erzeugen Sie eine neue Marke z.B. /tags/Version_1.0.1 aus dem Zweig an dem Sie gearbeitet haben.
- Wenn Sie eine Arbeitskopie, die auf einem Zweig basiert verändern und die Änderungen übertragen, gehen diese in den neuen Zweig und *nicht* in trunk. Nur die Unterschiede werden gespeichert. Der Rest bleibt eine billige Kopie.

4.20. Zusammenführen

Wenn Zweige für verschiedene Entwicklungslinien verwendet werden, werden Sie an einem bestimmten Punkt die Änderungen die Sie in einem Zweig gemacht haben wieder mit dem Stamm zusammenführen wollen, oder umgekehrt.

Es ist wichtig zu verstehen, wie Verzweigen und Zusammenführen in Subversion funktioniert bevor Sie es benutzen, denn es kann unter Umständen sehr komplex werden. Wir empfehlen Ihnen, das Kapitel *Branching and Merging* [<http://svnbook.red-bean.com/en/1.5/svn.branchmerge.html>] des Subversion Buchs zu lesen. Dieses Kapitel beschreibt ausführlich das Verzweigen und Zusammenführen und gibt auch einige Beispiele, wie man diese Werkzeuge am besten benutzt.

Der nächste wichtige Punkt ist, dass Zusammenführen *immer* in der Arbeitskopie stattfindet. Wenn Sie Änderungen *in einem Zweig* zusammenführen möchten, müssen Sie eine Arbeitskopie für diesen Zweig auschecken und den Assistenten mittels TortoiseSVN → Zusammenführen... in dieser Arbeitskopie aufrufen.

Es ist grundsätzlich eine gute Idee, Revisionen in einer unmodifizierten Arbeitskopie zusammenzuführen. Wenn Ihre Arbeitskopie Änderungen enthält, übertragen Sie diese zuerst. Falls nämlich das Zusammenführen nicht zu dem von Ihnen gewünschten Ergebnis führt, und sie die Änderungen rückgängig machen, wird dieser Befehl *alles*, inklusive Ihrer eigenen Änderungen verwerfen.

Es gibt drei Anwendungsfälle für das Zusammenführen, welche, wie weiter unten beschrieben, ein wenig unterschiedlich gehandhabt werden. Die erste Seite des Assistenten fragt Sie nach der gewünschten Methode.

Einen Revisionsbereich zusammenführen

Diese Methode wird benutzt wenn Sie Änderungen an einem Zweig (branch) oder Stamm (trunk) der Entwicklung gemacht haben und diese nun mit einem anderen Zweig (bzw. Stamm) zusammenführen möchten.

Damit weisen Sie Subversion an, folgendes zu tun: „Berechne die Änderungen, die nötig sind um *von* Revision X des Zweiges A *zu* Revision Y des Zweiges A zu gelangen und wende diese Änderungen auf meine Arbeitskopie (von Stamm oder Zweig B) an.“

Einen Zweig wieder eingliedern

Diese Methode behandelt den Fall dass Sie einen Funktionszweig, wie im Subversion Buch behandelt, angelegt haben. Alle Änderungen am Stamm wurden Woche für Woche mit dem Zweig zusammengeführt, und nun möchten Sie, da die Funktion im Zweig fertig ist, diese wieder im Stamm zusammenführen. Weil Sie den Funktionszweig immer mit dem Stamm synchronisiert haben, sind die letzten Versionen des Funktionszweiges und des Stammes, mit Ausnahme der auf dem Funktionszweig implementierten Neuerungen identisch.

Hierbei handelt es sich um einen Spezialfall des weiter unten beschriebenen Zusammenführens von Bäumen. Es wird als Quelle nur die URL des Entwicklungszweiges benötigt. Die Funktion verwendet die Protokollierung der Datenintegration, um die korrekten Revisionsbereiche zu ermitteln und führt zusätzlich Überprüfungen durch ob der Zweig mit allen Änderungen aus dem Stamm zusammengeführt wurde. Dadurch wird sichergestellt, dass Sie nicht versehentlich Änderungen von anderen zurücknehmen, die seit Ihrer letzten Synchronisation mit dem Stamm übertragen wurden.

Nach dem Zusammenführen ist der Funktionszweig komplett in die Hauptentwicklungslinie zurückgeführt worden. Der Zweig ist nun überflüssig und kann gelöscht werden.

Sobald Sie einen Zweig wieder eingliedert haben, sollten Sie ihn nicht mehr für die Weiterentwicklung nutzen. Der Grund ist, dass wenn Sie ihren existierenden Zweig später mit dem Stamm synchronisieren wollen, die Änderungsverfolgung ihre Wiedereingliederung als Änderung am Stamm sieht, die noch nicht in den Zweig übertragen wurde und versucht, die Zweig nach Stamm Integration zurück in den Zweig zu integrieren! Die Lösung für dieses Problem besteht darin, für die nächste Phase der Weiterentwicklung einen neuen Zweig vom Stamm aus anzulegen.

Zusammenführen zweier Bäume

Dies ist ein allgemeinerer Fall des wieder Eingliederns eines Zweiges. Damit weisen Sie Subversion an, folgendes zu tun: „Berechne die Änderungen, die nötig sind um *von* der HEAD Revision des Stammes *zur* HEAD Revision des Zweiges zu gelangen und wende diese Änderungen auf meine Arbeitskopie (des Stammes) an.“ Das Ergebnis ist, dass der Stamm danach exakt wie der Zweig aussieht.

Falls Ihr Server oder das Projektarchiv die Protokollierung der Datenintegration nicht unterstützen, ist das die einzige Methode mit der Sie einen Zweig in den Stamm zurückintegrieren können. Ein anderer Anwendungsfall tritt auf, wenn Sie mit Herstellerzweigen arbeiten und die Änderungen aufgrund einer neuen Auslieferung in Ihrem Stamm zusammenführen müssen. Für weiterführende Informationen lesen Sie bitte das Kapitel *vendor branches* [<http://svnbook.red-bean.com/en/1.5/svn.advanced.vendorbr.html>] im Subversion Buch.

4.20.1. Einen Revisionsbereich zusammenführen

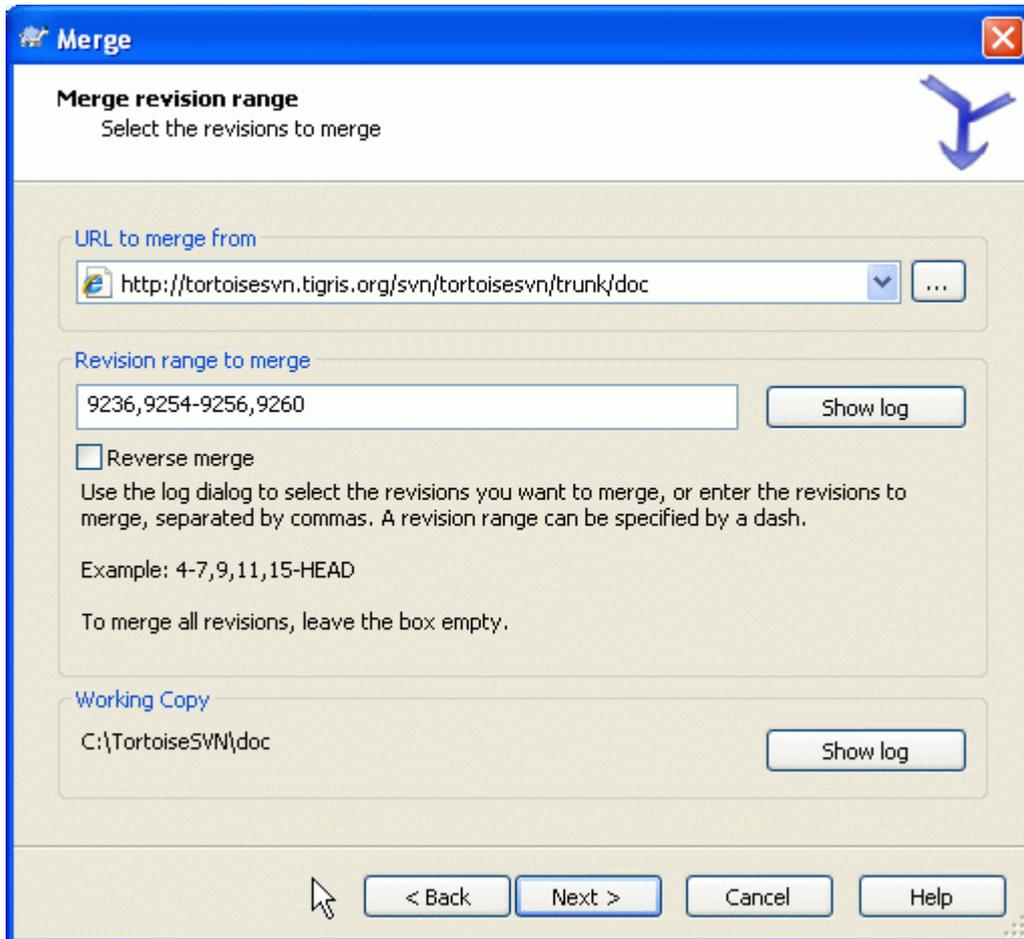


Abbildung 4.36. Der Assistent - Revisionsbereich wählen

Im Von: Feld geben Sie die URL des Zweiges an, der die Änderungen enthält welche Sie mit der Arbeitskopie zusammenführen möchten. Sie können auch auf ... klicken, um den gewünschten Zweig zu finden. Wenn Sie bereits Änderungen aus diesem Zweig zusammengeführt haben, finden Sie den Namen in der Liste der bereits benutzten URLs.

Geben Sie den zusammenzuführenden Revisionsbereich in das Feld Revisionsbereich ein. Das kann eine einzelne Revision, eine durch Kommata getrennte Liste von Revisionen, ein durch Bindestrich getrennter Revisionsbereich oder eine Kombination aus allem sein.



Wichtig

Es gibt einen wichtigen Unterschied in der Spezifikation von Revisionsbereichen, wie sie in TortoiseSVN und im Kommandozeilenclient verwendet werden. Am einfachsten stellt man sich das als einen Zaun mit Pfählen und Brettern dazwischen vor.

Mit dem Kommandozeilenclient legt man die Revisionen der zusammenzuführenden Bereiche als „Zaunpfähle“ fest, die die Punkte *davor* und *danach* festlegen.

Mit TortoiseSVN legen Sie die zusammenzuführenden Änderungen als „Zaubretter“ fest. Der Grund dafür wird deutlich, wenn Sie den Log-Dialog verwenden, um die Revisionen festzulegen, in dem jede Revision als ein Satz von Änderungen erscheint.

Wenn Sie Revisionen in Blöcken zusammenführen, würde die Methode im Subversion Buch sie zunächst 100-200 und danach 200-300 zusammenführen lassen. Mit TortoiseSVN sind es zunächst 100-200 und dann 201-300.

Dieser Unterschied hat zu hitzigen Diskussionen auf der Mailingliste geführt. Wir räumen ein, dass es einen Unterschied zum Kommandozeilenclient gibt, aber wie sind davon überzeugt, dass die von uns implementierte Methode für die mehrzahl der GUI Anwender einfacher zu verstehen ist.

Der einfachste Weg um den Revisionsbereich zu wählen, ist ein Klick auf **Zeige Log**. Dies zeigt Ihnen die letzten Änderungen inklusive der Logmeldungen an. Wenn Sie nur die Änderungen einer einzelnen Revision benötigen, wählen Sie die gewünschte Revision. Wenn Sie einen Revisionsbereich benötigen, wählen Sie diesen mit gedrückter **Umsch** Taste aus. Klicken Sie auf **OK** und die Liste der zusammenzuführenden Revisionsnummern wird für Sie ausgefüllt.

Wenn Sie bereits übertragene Änderungen rückgängig machen wollen, erreichen Sie das, indem Sie den gewünschten Revisionsbereich **Rückwärts zusammenführen**. Dadurch werden die Änderungen dieses Revisionsbereichs aus Ihrer Arbeitskopie entfernt.

Wenn Sie bereits Änderungen aus diesem Zweig zusammengeführt haben, so haben Sie hoffentlich auch in der Logmeldung notiert, welche Revisionen das waren. In diesem Fall können Sie einfach mit Hilfe der **Zeige Log** Schaltfläche nachschauen. Verwenden Sie die Endrevision des letzten Zusammenführens als Startrevision der geplanten Aktion. Wenn Sie zum Beispiel zuletzt die Revisionen 37 bis 39 mit Ihrer Arbeitskopie zusammengeführt haben, sollte die Startrevision für die aktuelle Aktion 40 sein.

Neuere Versionen von Subversion protokollieren die bisher durchgeführten Datenintegrationen mit. Deshalb müssen Sie sich nicht merken, welche Revisionen Sie bereits zusammengeführt haben. Wenn Sie den Revisionsbereich leer lassen, werden alle bisher noch nicht integrierten Revisionen ausgewählt. Lesen Sie in [Abschnitt 4.20.6, „Verfolgung der Datenintegration“](#) wie das funktioniert.

Falls außer Ihnen noch andere Personen Daten in das Projektarchiv übertragen, seien Sie vorsichtig mit der Verwendung der HEAD Revision. Diese entspricht eventuell nicht der Revision die Sie erwarten, weil seit Ihrer letzten Aktualisierung jemand anderes Daten übertragen haben könnte.

Klicken Sie auf **Weiter** und gehen Sie zu [Abschnitt 4.20.4, „Optionen beim Zusammenführen“](#)

4.20.2. Einen Zweig wieder eingliedern

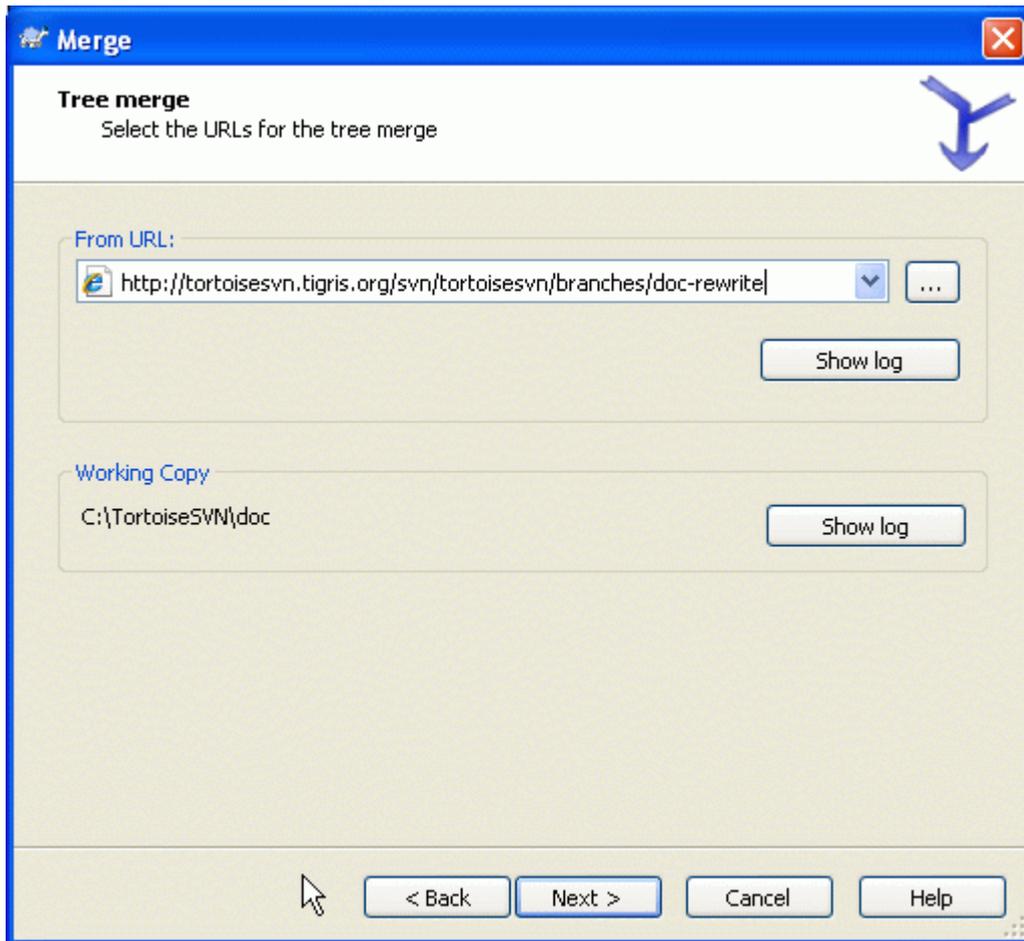


Abbildung 4.37. Der Assistent - Zweig wieder Eingliedern

Um einen Funktionszweig mit dem Stamm zusammenzuführen, müssen Sie den Assistenten aus einer Arbeitskopie des Stammes heraus aufrufen

Im Von: Feld geben Sie die vollständige URL des *Zweiges* an, den Sie zurückintegrieren möchten. Sie können auch auf ... klicken um die URL besser zu finden.

Um einen Zweig wieder Eingliedern zu können, müssen einige Bedingungen erfüllt sein. Zunächst muss der Server die Verfolgung der Datenintegration unterstützen. Die Tiefe der Arbeitskopie muss unendlich sein (keine dünn besiedelte Arbeitskopie) und sie darf weder lokalen Änderungen noch Objekte auf einer anderen URL oder einer anderen Revision als HEAD enthalten. Alle Änderungen am Stamm während der Entwicklung des Zweiges müssen in den Zweig übertragen worden sein (oder als übertragen markiert worden sein). Der Bereich der zusammenzuführenden Revisionen wird dann automatisch berechnet.

4.20.3. Zusammenführen zweier Bäume

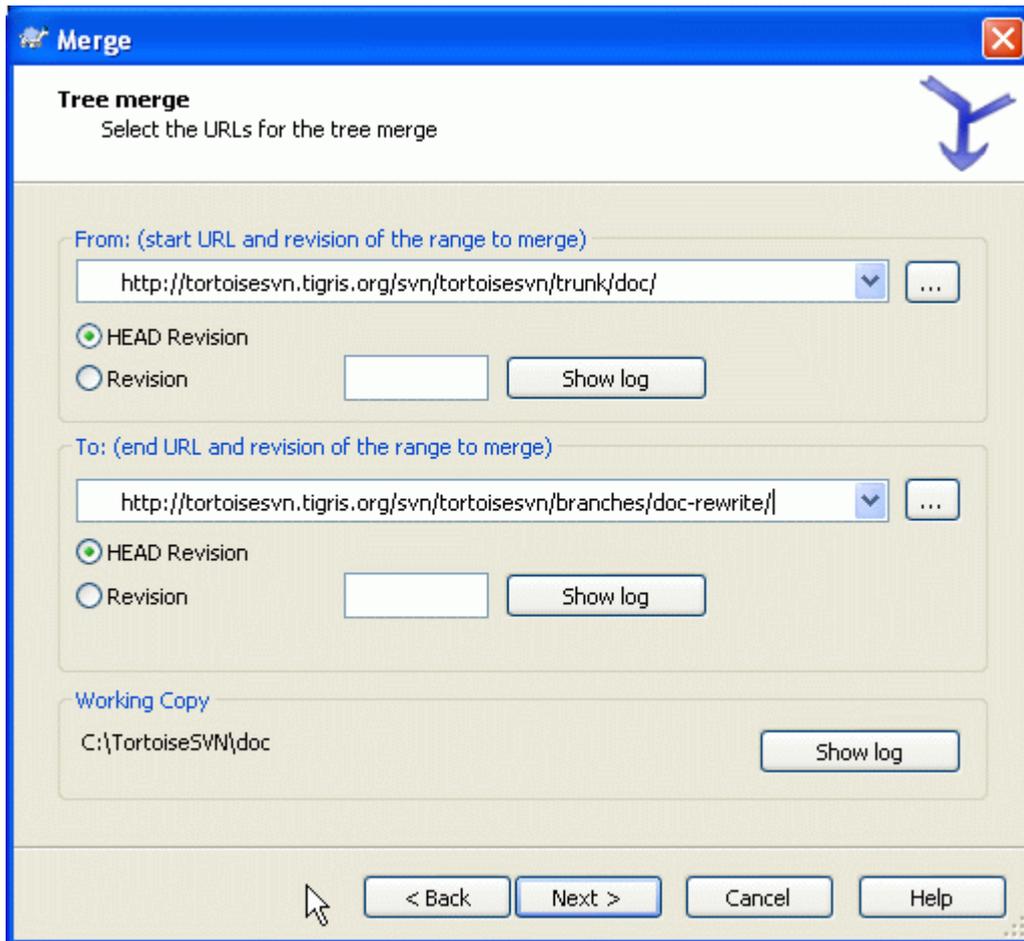


Abbildung 4.38. Der Assistent - Zusammenführen von Bäumen

Um mit dieser Methode einen Funktionszweig mit dem Stamm zusammenzuführen, müssen Sie den Assistenten aus einer Arbeitskopie des Stamms starten.

Im Von: Feld geben Sie die URL des *Stammes (trunk)* an. Dies mag Ihnen jetzt vielleicht falsch vorkommen, aber bedenken Sie dass der Stamm der Ausgangspunkt ist von dem aus Sie die Änderungen des Funktionszweiges zusammenführen möchten. Sie können auch auf ... klicken um die URL besser zu finden.

Im Bis: Feld geben Sie die vollständige URL des Zweiges an.

Sowohl im Von Revision Feld als auch im Bis Revision Feld geben Sie die letzte Revision an in der die beiden Zweige synchron waren. Wenn Sie sicher sind dass niemand anders seit Ihrer letzten Synchronisation eine Übertragung gemacht hat, können Sie in beiden Feldern die HEAD Revision angeben. Falls jedoch in der Zwischenzeit die Möglichkeit einer Übertragung besteht, geben Sie die letzte Revision Ihrer Synchronisation an.

Sie können auch auf Zeige Log klicken, um die Revision zu wählen.

4.20.4. Optionen beim Zusammenführen

Auf dieser Seite des Assistenten können Sie weitere Optionen angeben, bevor Sie das Zusammenführen starten. In den meisten Fällen können Sie die Standardeinstellungen verwenden.

Sie können die Tiefe für das Zusammenführen festlegen, das heißt, wie weit das Zusammenführen in Ihre Arbeitskopie herabsteigen soll. Die Tiefendefinitionen sind in [Abschnitt 4.3.1, „Rekursionstiefe“](#)

beschrieben. Die Vorgabe ist **Arbeitskopie**, was den aktuell eingestellten Wert verwendet und meistens korrekt sein wird.

Meistens möchten Sie, dass die Vergangenheit von Dateien berücksichtigt wird, so dass Änderungen relativ zu einem gemeinsamen Vorgänger zusammengeführt werden. Manchmal möchten Sie aber auch Dateien zusammenführen, zwischen denen eine Beziehung besteht - allerdings nicht in Ihrem Projektarchiv. Sie haben zum Beispiel Versionen 1 und 2 einer externen Bibliothek in zwei verschiedene Verzeichnisse importiert. Obwohl zwischen diesen ein logischer Zusammenhang besteht, weiß Subversion nichts davon, weil es nur die separat importierten Dateibäume sieht. Wenn Sie versuchen, die Unterschiede zwischen den beiden Versionen zusammenzuführen, werden Sie ein vollständiges Entfernen gefolgt von einem vollständigen Hinzufügen erhalten. Damit Subversion nur pfadbasierte Differenzen anstelle von vergangenheitsbasierten Differenzen erzeugt, wählen Sie die **Vorfahren ignorieren** Option. Weitere Informationen zu diesem Thema finden Sie im Subversion Buch in *Noticing or Ignoring Ancestry* [<http://svnbook.red-bean.com/en/1.5/svn.branchmerge.advanced.html#svn.branchmerge.advanced.ancestry>]

Sie können festlegen, wie Änderungen an Zeilenumbrüchen und Leerzeichen behandelt werden sollen. Diese Optionen sind in **Abschnitt 4.10.2, „Zeilende- und Leerzeichenoptionen“** beschrieben. Die Vorgabe ist, dass alle Zeilenenden- und Leerzeichenänderungen als echte, zusammenzuführende Änderungen behandelt werden.

Falls Sie Ihre Datenintegrationsschritte verfolgen und eine Revision als zusammengeführt markieren wollen, ohne die Daten wirklich zusammenzuführen, aktivieren Sie die Option **Zusammenführen nur aufzeichnen**. Es gibt mehrere Gründe, dies zu tun. Zum Beispiel könnten die zusammenzuführenden Änderungen zu komplex für den Algorithmus sein, so dass Sie die Daten von Hand integrieren und die Revisionen als zusammengeführt markieren, damit die Änderungsverfolgung darüber informiert ist. Oder Sie möchten verhindern, dass eine bestimmte Revision zusammengeführt werden kann. Clients die die Verfolgung der Datenintegration bereits beherrschen, werden diese Revision dann überspringen.

Nun, da alles vorbereitet ist, müssen Sie nur noch auf **Zusammenführen** klicken. Falls Sie eine Vorschau der Ergebnisse wünschen, können Sie einen **Trockenlauf** durchführen, der das Zusammenführen lediglich simuliert und die Arbeitskopie *nicht* verändert. Eine Liste der veränderten Dateien und der eventuell resultierenden Konflikte wird angezeigt.

Der Fortschrittsdialog zeigt nun jeden Schritt des Zusammenführens mit den betroffenen Revisionen an. Eventuell wird hier eine Revision mehr, als von Ihnen erwartet, angezeigt. So wird zum Beispiel, falls Sie die Revision 123 zusammenführen, im Fortschrittsdialog „Zusammenführen von Revisionen 122 bis 123“ angezeigt. Das Zusammenführen ist mit dem Vergleichen verwandt. Der Prozess erzeugt eine Liste von Änderungen zwischen zwei Punkten im Projektarchiv und wendet diese Änderungen auf Ihre Arbeitskopie an. Der Fortschrittsdialog zeigt lediglich den Start- und Endpunkt der Differenzbildung an.

4.20.5. Ergebnisse des Zusammenführens betrachten

Das Zusammenführen ist nun beendet. Es ist eine gute Idee, wenn Sie sich das Ergebnis anschauen und prüfen, ob es Ihren Erwartungen entspricht. Datenintegration ist normalerweise eine komplizierte Sache. Konflikte können auftreten, wenn der Zweig sich zu weit vom Stamm entfernt hat.

Bei Subversion Clients vor Version 1.5 müssen Informationen über zusammengeführte Revisionen von Hand protokolliert werden. Wenn Sie die Änderungen überprüft haben und diese nun übertragen möchten, beachten Sie dass die Logmeldung dieser Übertragung *immer* die bereits zusammengeführten Revisionsnummern enthalten sollte. Wenn Sie später erneut Revisionen zusammenführen möchten/müssen werden Sie froh sein zu wissen, was alles Sie bereits zusammengeführt haben und was nicht. Denn wenn Sie eine Revision ein weiteres Mal zusammenführen, würde dies zwingend zu einem Konflikt führen. Für genauere Informationen lesen Sie bitte *Best Practices for Merging* [<http://svnbook.red-bean.com/en/1.4/svn.branchmerge.copychanges.html#svn.branchmerge.copychanges.bestprac>] im Subversion Buch.

Wenn Ihr Server und alle Clients unter Subversion 1.5 oder neuer laufen, wird die Protokollierung der Datenintegration sämtliche zusammengeführten Revisionen aufzeichnen und verhindern, dass eine Revision mehr als einmal integriert wird. Das vereinfacht Ihre Arbeit ungemein, da sie nun jedes Mal

den gesamten Revisionsbereich wählen können und sichergestellt ist, dass nur neue Revisionen wirklich zusammengeführt werden.

Das Verwalten von Zweigen ist wichtig. Falls Sie Ihren Zweig stets mit dem Stamm synchron halten wollen, stellen Sie sicher, dass Sie Änderungen häufig zusammenführen, damit Stamm und Zweig nicht zu weit auseinanderdriften. Wie bereits erklärt sollten Sie wiederholtes Zusammenführen der gleichen Änderungen vermeiden.



Tipp

Wenn Sie gerade einen Funktionszweig in den Stamm integriert haben, enthält der Stamm nun sämtliche neuen Funktionen. Der Zweig ist nunmehr überflüssig und kann deshalb gelöscht werden.



Wichtig

Subversion kann keine Datei mit einem Ordner zusammenführen und umgekehrt - nur Ordner mit Ordner und Dateien mit Dateien können zusammengeführt werden. Wenn Sie auf eine Datei klicken um den Zusammenführen-Dialog zu öffnen, müssen Sie eine URL zu einer Datei angeben, wenn Sie auf einen Ordner klicken um den Zusammenführen-Dialog zu öffnen müssen Sie für den Vorgang eine URL zu einem Ordner angeben.

4.20.6. Verfolgung der Datenintegration

Mit Subversion 1.5 wurden Möglichkeiten zur Verfolgung der Datenintegration eingeführt. Wenn Sie Änderungen von einem Baum in einem anderen zusammenführen, werden die entsprechenden Revisionsnummern gespeichert. Diese Information kann für weitere Zwecke herangezogen werden.

- Sie können vermeiden, dass eine Revision mehrfach integriert wird. Sobald eine Revision als bereits zusammengeführt markiert ist, werden zukünftige Datenintegrationen diese Revision überspringen.
- Wenn Sie einen Zweig in den Stamm zurückintegrieren, kann Ihnen der Log-Dialog die Übertragungen des Zweiges im Log des Stamms anzeigen und Ihnen damit die Verfolgung der Änderungen vereinfachen.
- Wenn Sie den Log-Dialog aus dem Zusammenführen-Dialog heraus aufrufen, werden bereits zusammengeführte Revisionen in grau angezeigt.
- Wenn Sie die Annotierungen für eine Datei anzeigen, können Sie sich den Originalautor der Änderungen anstelle des Autors, der die Änderungen zusammengeführt hat, anzeigen lassen.
- Sie können Revisionen als *nicht zusammenführbar* markieren, indem Sie diese in die Liste der bereits zusammengeführten Revisionen aufnehmen, ohne das Zusammenführen tatsächlich anzustoßen.

Die zusammengeführten Revisionen und Pfade werden durch den Subversion Client beim Zusammenführen in der `svn:mergeinfo` Eigenschaft abgespeichert. Sobald die Änderungen übertragen wurden, speichert der Server sie in einer Datenbank. Falls jetzt Logdaten, Annotierungen oder Zusammenführen angefordert werden, kann der Server entsprechend reagieren. Damit das System korrekt arbeitet, müssen Sie sicherstellen, dass der Server, das Projektarchiv und alle Clients aktualisiert werden. Ältere Clients füllen die `svn:mergeinfo` nicht aus und ältere Server beantworten die Anfragen neuerer Clients nicht.

Lesen Sie mehr über die Verfolgung der Datenintegration in Subversions [Merge tracking documentation](http://subversion.tigris.org/merge-tracking/index.html) [http://subversion.tigris.org/merge-tracking/index.html].

4.20.7. Behandeln von Konflikten beim Zusammenführen

Zusammenführen geht nicht immer glatt. Manchmal gibt es einen Konflikt, und wenn Sie mehrere Bereiche zusammenführen, möchten Sie normalerweise die Konflikte auflösen, bevor Sie den nächsten Bereich angehen. TortoiseSVN hilft Ihnen durch diesen Prozess, indem es den *Der Rückfrage-Dialog für Konflikte* anzeigt.

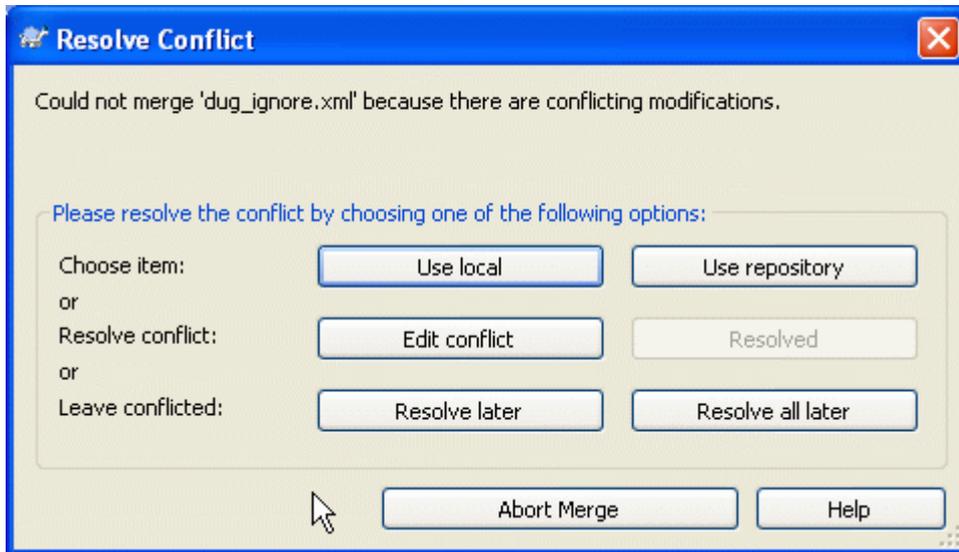


Abbildung 4.39. Der Rückfrage-Dialog für Konflikte

Wenn beim Zusammenführen ein Konflikt auftritt, haben Sie drei Möglichkeiten diesen aufzulösen:

1. Sie können entscheiden, dass Ihre lokalen Änderungen wichtiger sind und die Version aus dem Projektarchiv verwerfen, um die lokale Version zu behalten. Oder sie verwerfen Ihre lokalen Änderungen zugunsten der Serverversion. Welche Option Sie auch immer wählen, es werden nie Daten zusammengeführt - Sie erhalten eine der beiden Dateien.
2. Normalerweise werden Sie die Konflikte ansehen und beseitigen wollen. In diesem Fall wählen Sie **Konflikt bearbeiten**, wodurch Ihr Konflikteditor gestartet wird. Sobald Sie mit dem Ergebnis zufrieden sind, wählen Sie **Aufgelöst**.
3. Die letzte Option ist, das Auflösen der Konflikte zu verschieben und mit dem Zusammenführen fortzufahren. Sie können das für die aktuelle Datei oder für alle weiteren Dateien beim Zusammenführen festlegen. Wenn es jedoch weitere Änderungen in dieser Datei gibt, können Sie das Zusammenführen nicht abschließen.

Wenn Sie den interaktiven Rückfrage-Dialog nicht benutzen wollen, können sie die Funktion über die Option **Nicht interaktiv zusammenführen** im Fortschrittsdialog abschalten. Sobald der Dialog deaktiviert ist, werden Dateien mit Konflikten beim Zusammenführen markiert und Sie müssen die Konflikte auflösen *nachdem* das gesamte Zusammenführen beendet ist. Wenn das interaktive Zusammenführen aktiv ist, erhalten Sie die Gelegenheit, einen Konflikt *während* des Zusammenführens aufzulösen, bevor die Datei als konfliktbehaftet markiert wird. Dadurch können, wenn mehrere zusammenzuführende Revisionen nacheinander auf eine Datei angewendet werden, spätere Aktionen erfolgreich sein. Aber leider können Sie in diesem Fall nicht weggehen, um sich einen Kaffee zu holen während das Zusammenführen läuft.

4.20.8. Einen vollständigen Zweig zusammenführen

Wenn Sie sämtliche Änderungen des Zweiges, an dem Sie gerade arbeiten in den Stamm, zurück übertragen möchten, können Sie TortoiseSVN → **Erneut zusammenführen...** aus dem erweiterten Kontextmenü aufrufen (Halten Sie die **Umsch** Taste gedrückt, während Sie einen Rechtsklick auf die Datei machen).

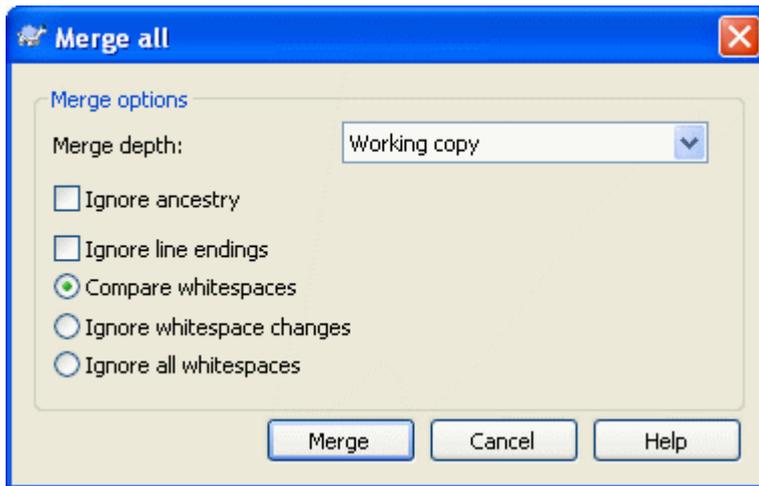


Abbildung 4.40. Der wieder Eingliedern Dialog

Dieser Dialog ist sehr einfach. Sie müssen lediglich die in [Abschnitt 4.20.4, „Optionen beim Zusammenführen“](#) beschriebenen Optionen einstellen. Der Rest wird mit Hilfe der protokollierten Integrationen automatisch von TortoiseSVN erledigt.

4.20.9. Wartung des Funktionszweiges

Wenn Sie eine neue Funktion in einem separaten Zweig entwickeln, ist es eine gute Idee, sich rechtzeitig eine Richtlinie zu überlegen, wie diese Funktion in die Hauptentwicklungslinie zurückintegriert wird. Falls gleichzeitig in `trunk` weiter entwickelt wird, können die Unterschiede mit der Zeit bedeutend werden und die Rückintegration zu einem Alptraum machen.

Falls die Funktion relativ einfach ist und keine längere Zeit in Anspruch nehmen wird, können Sie den einfachen Ansatz wählen und auf dem Zweig so lange separat entwickeln, bis die Funktion komplett ist und dann zurück integrieren. Im Assistenten wäre dies eine einfache `Einen Revisionsbereich zusammenführen` Aktion, mit dem Revisionsbereich des Zweiges als Parameter.

Falls die Entwicklung länger dauern wird und Sie Änderungen in `trunk` berücksichtigen müssen, sollten Sie den Zweig synchron halten. Das bedeutet dass Sie periodisch Änderungen an `trunk` in den Zweig integrieren, so dass der Zweig alle Änderungen an `trunk` *plus* die neue Funktion enthält. Beim Synchronisationsvorgang verwenden Sie `Einen Revisionsbereich zusammenführen`. Sobald die Funktion komplett ist, können Sie die Änderungen entweder per `Einen Zweig wieder Eingliedern` oder `Zwei verschiedene Bäume zurück integrieren`.

4.21. Sperren

Subversion funktioniert im Allgemeinen am besten ohne Sperren, da es den in [Abschnitt 2.2.3, „Die Kopieren-Ändern-Zusammenführen Lösung“](#) besprochenen Ansatz verfolgt. Es gibt jedoch einige Fälle, in denen es erforderlich ist, Dateien zu sperren.

- Sie arbeiten mit „nicht zusammenführbaren“ Dateien, z.B. Grafiken. Wenn zwei Personen dieselbe Datei modifizieren, ist ein Zusammenführen nicht möglich, so dass einer der beiden seine Änderungen verlieren wird.
- Ihre Firma hat bisher immer ein sperrendes Versionskontrollsystem eingesetzt und es gibt eine Management Entscheidung, dass „Sperren das Beste“ ist.

Zunächst müssen Sie sicherstellen, dass Ihr Subversion Server zumindest mit Version 1.2 arbeitet. Frühere Versionen unterstützen Sperren überhaupt nicht. Wenn Sie ausschließlich den `file://` Zugriff nutzen, müssen nur Ihre Clients aktualisiert werden.

4.21.1. Sperren von Dateien in Subversion

In der Grundeinstellung sind keine Objekte gesperrt und jeder, der die entsprechende Berechtigung hat, kann seine Änderungen in das Projektarchiv übertragen. Andere werden ihre Arbeitskopien regelmäßig aktualisieren und Änderungen im Projektarchiv werden mit den lokalen Änderungen zusammengeführt.

Wenn Sie für eine Datei *Eine Sperre holen*, können nur Sie Änderungen an dieser Datei übertragen. Übertragungen von anderen werden so lange verhindert, bis Sie die Sperre wieder freigeben. Eine gesperrte Datei kann auf keine Weise im Projektarchiv verändert, umbenannt oder gelöscht werden. Dies ist nur dem Eigner der Sperre möglich.

Andere Anwender werden nicht unbedingt wissen, dass Sie eine Datei gesperrt haben. Wenn sie nicht regelmäßig den Sperrstatus überprüfen, werden sie es frühestens merken, wenn ihre Übertragung fehlschlägt, was in den meisten Fällen nicht sehr nützlich ist. Damit es einfacher ist, Sperren zu verwalten, gibt es die neue `svn:needs-lock` Eigenschaft von Subversion. Wenn diese Eigenschaft (auf einen beliebigen Wert) gesetzt ist, wird beim Auschecken oder Aktualisieren die Datei in der Arbeitskopie mit einem Schreibschutz versehen, *es sei denn* die Arbeitskopie besitzt die Sperre für die Datei. Dies dient als Warnung, dass Sie die Datei nicht bearbeiten sollen, bevor Sie nicht die Sperre für die Datei besitzen. Dateien unter Versionskontrolle, die mit Schreibschutz versehen sind, erhalten von TortoiseSVN ein überlagertes Symbol, das anzeigt, dass Sie erst eine Sperre holen müssen, bevor Sie die Datei bearbeiten.

Sperren gehören sowohl zu einer Arbeitskopie als auch zu einer Person. Wenn Sie mehrere Arbeitskopien desselben Projekts (bei der Arbeit, daheim) haben, können Sie eine Datei nur in *einer* dieser Arbeitskopien sperren.

Wenn einer Ihrer Kollegen eine Datei sperrt und in Urlaub geht, ohne die Sperre vorher freizugeben, was dann? Subversion bietet Ihnen eine Möglichkeit, Sperren auszuhebeln. Eine Sperre freizugeben, die jemand anders besitzt, wird auch *Sperre aufbrechen* genannt. Sich eine Sperre zu holen, die jemand anders besitzt, wird auch *Sperre stehlen* genannt. Selbstverständlich sollten Sie dies nicht leichtfertig tun, wenn Sie sich weiterhin mit Ihren Kollegen gut stellen wollen.

Sperren werden im Projektarchiv verwaltet und eine Sperrmarke wird zusätzlich in Ihrer Arbeitskopie angelegt. Wenn es einen Unterschied zwischen diesen Werten gibt, z.B. weil jemand eine Sperre aufgebrochen hat, wird die lokale Sperrmarke ungültig. Der Sperrstatus im Projektarchiv ist stets maßgeblich.

4.21.2. Eine Sperre erhalten

Markieren Sie die Datei(en) in Ihrer Arbeitskopie, für die Sie eine Sperre erhalten möchten und wählen Sie TortoiseSVN → Sperre holen....

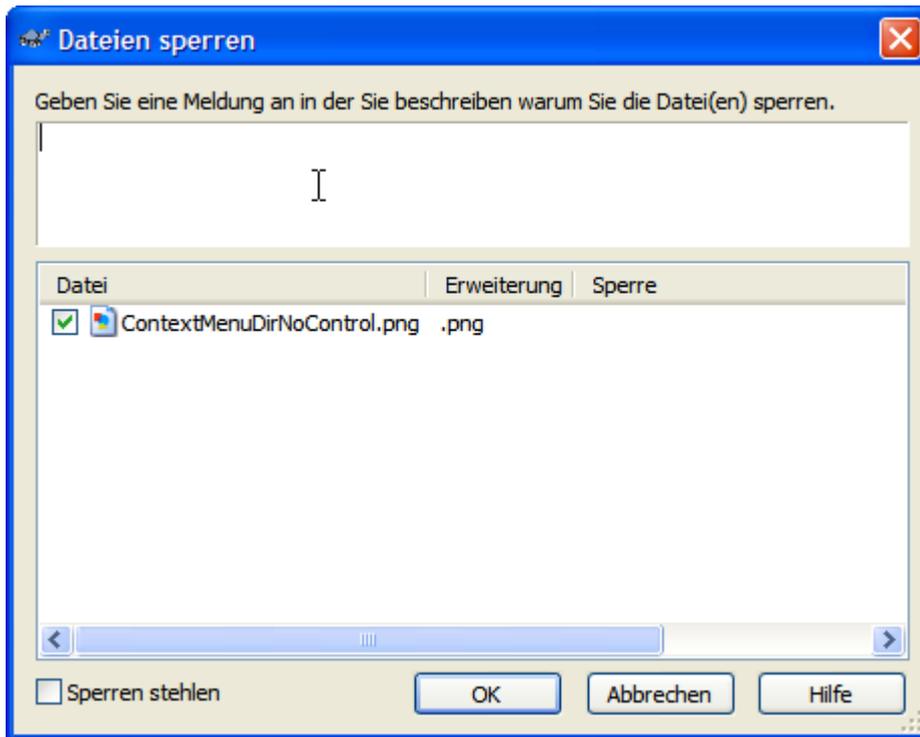


Abbildung 4.41. Der Sperren-Dialog

Ein Dialog erscheint, in dem Sie einen Sperrkommentar eingeben können, damit andere wissen, warum Sie die Datei für sich reservieren. Der Kommentar ist optional und wird derzeit nur bei svnserve basierten Projektarchiven genutzt. Wenn (und *nur* wenn) sie eine Sperre von jemand anderem stehlen wollen, aktivieren Sie die Option **Sperrung stehlen** bevor Sie auf **OK** klicken.

Wenn Sie einen Ordner markieren und TortoiseSVN → **Sperre Holen...** aufrufen, wird der Sperr-Dialog mit *jeder* Datei in *jedem* Unterordner, zum Sperren markiert, geöffnet. Wenn Sie wirklich eine ganze Ordnerstruktur sperren wollen, ist das der richtige Weg das zu erreichen. Sie können sich damit allerdings bei Ihren Kollegen sehr unbeliebt machen, wenn Sie diese aus dem gesamten Projekt aussperren...

4.21.3. Eine Sperre freigeben

Um sicherzustellen, dass Sie nicht vergessen, Sperren freizugeben, werden gesperrte Dateien im Übertragen-Dialog angezeigt und sind standardmäßig gewählt. Wenn Sie die Übertragung durchführen, werden die Sperren der markierten Dateien freigegeben, auch wenn die Dateien selbst sich nicht geändert haben. Wenn Sie die Dateisperren nicht freigeben wollen, können Sie die entsprechenden Dateien abwählen, vorausgesetzt, dass sie nicht modifiziert sind. Wenn Sie die Sperre für veränderte Dateien behalten wollen, müssen Sie vor der Übertragung die **Sperren behalten** Option markieren.

Um eine Sperre von Hand freizugeben, markieren Sie die gewünschten Datei(en) in Ihrer Arbeitskopie und wählen Sie TortoiseSVN → **Sperre freigeben**. TortoiseSVN wird das Projektarchiv kontaktieren und die Sperren dort freigeben. Sie können diesen Befehl auch für einen Ordner aufrufen, um alle Sperren rekursiv freizugeben.

4.21.4. Den Sperrstatus prüfen

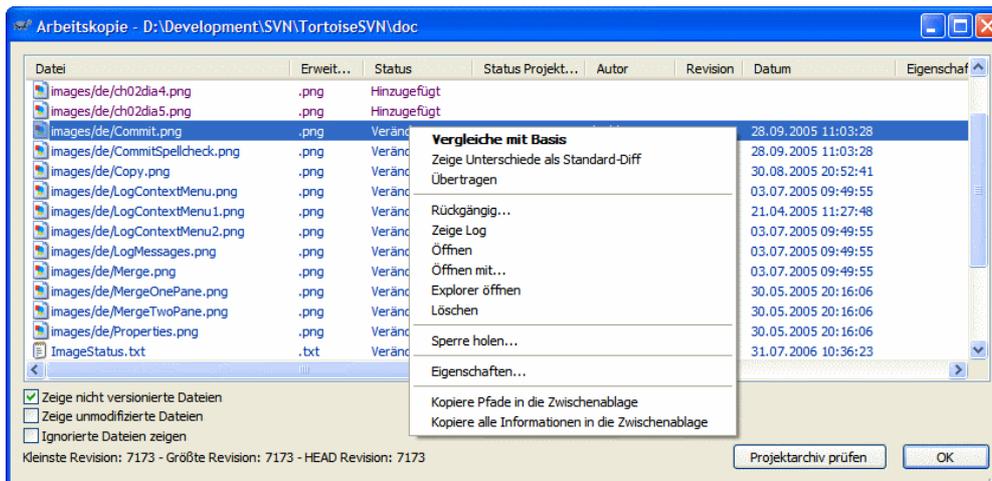


Abbildung 4.42. Der Dialog „Prüfe auf Änderungen“

Um zu sehen, welche Sperren Sie oder andere Personen besitzen, können Sie TortoiseSVN → Prüfe auf Änderungen... aufrufen. Lokale Sperren werden sofort angezeigt. Damit Sie sehen können, welche Sperren von anderen gehalten werden und um festzustellen, ob Ihre Sperren aufgebrochen oder gestohlen wurden, müssen Sie Projektarchiv prüfen wählen.

Aus dem Kontextmenü dieses Dialogs heraus können Sie Sperren holen oder freigeben und auch Sperren von anderen stehlen oder aufbrechen.



Vermeiden Sie es Sperren aufzubrechen oder zu stehlen

Wenn Sie die Sperre eines Kollegen stehlen oder aufbrechen, ohne ihm dies mitzuteilen, können Sie doppelte Arbeit verursachen. Falls Sie mit nicht zusammenführbaren Dateien arbeiten und eine Sperre stehlen, kann derjenige der die Sperre vorher besaß Ihre Daten überschreiben, sobald Sie die Sperre freigeben. Subversion verliert zwar keine Daten, aber Sie haben den gegenseitigen Schutz bei der Zusammenarbeit, den Sperren bieten, verloren.

4.21.5. Nicht gesperrte Dateien mit Schreibschutz versehen

Wie bereits erwähnt ist der einfachste Weg dazu, die `svn:needs-lock` Eigenschaft der Datei zu setzen. Lesen Sie [Abschnitt 4.17, „Projekt-Einstellungen“](#), um zu erfahren, wie man Eigenschaften setzt. Dateien mit dieser Eigenschaft werden beim Aktualisieren und Auschecken stets mit einem Schreibschutz versehen, wenn die Arbeitskopie keine Sperre für die Datei besitzt.



Als Erinnerung zeigt TortoiseSVN dieses Symbol dafür an.

Wenn Sie ein Entwicklungsverfahren nutzen, das stets Dateisperren erfordert, ist es einfacher Subversions `auto-props` Eigenschaft zu nutzen, um die Dateieigenschaften automatisch zu setzen, sobald Sie Dateien zum Projektarchiv hinzufügen. Lesen Sie [Abschnitt 4.17.1.5, „Eigenschaften automatisch setzen“](#) für weitere Information.

4.21.6. Aktionsskripte für Sperren

Wenn Sie ein neues Projektarchiv mit Subversion 1.2 anlegen, werden vier Schablonen für Aktionsskripte im `hooks` Verzeichnis des Projektarchivs angelegt. Diese Aktionsskripte werden aufgerufen, bevor und nachdem eine Sperre geholt, bzw. freigegeben wird.

Es ist eine gute Idee, ein `post-lock` und ein `post-unlock` Aktionsskript auf dem Server zu installieren, die eine Benachrichtigungs E-Mail verschicken. Mit solch einem Skript können alle betroffenen Personen darüber informiert werden, wenn eine Datei gesperrt wurde. Sie finden ein Beispiel `hooks/post-lock.tmp1` Skript in Ihrem Projektarchiv.

Vielleicht möchten Sie mit Aktionsskripten auch verhindern, dass jemand Sperren stiehlt oder aufbricht oder Sie möchten diese Möglichkeit auf Administratoren beschränken. Sie können mit einem Skript auch den Eigner der Sperre per E-Mail darüber informieren, wenn seine Sperre gestohlen wurde.

Lesen Sie [Abschnitt 3.3, „Serverseitige Aktionsskripte“](#) für weitere Informationen.

4.22. Erzeugen und Anwenden von Patches

In Open Source Projekten (wie diesem) hat jedermann Lesezugriff auf das Projektarchiv, und jeder kann einen Beitrag zum Projekt leisten. Wie sollen diese Beiträge nun kontrolliert werden? Wenn jeder stets seine Änderungen in das Projektarchiv übertragen würde, wäre das Projekt permanent instabil und wahrscheinlich sogar nicht funktionsfähig. Dieses Problem wird gelöst, indem ein *Patch* an das Entwicklerteam geschickt wird. Die Entwickler können den Patch überprüfen und dann entweder in das Projektarchiv übertragen oder verwerfen und an den Autor zurückschicken.

Bei Patchdateien handelt es sich um einfache Standard-Diff Dateien, die die Unterschiede zwischen Ihrer Arbeitskopie und der Basisrevision beinhalten.

4.22.1. Eine Patch-Datei erstellen

Zunächst müssen Sie Ihre Änderungen selbstverständlich selber *testen*. Anstatt den übergeordneten Ordner zum Projektarchiv zu TortoiseSVN → Übertragen..., wählen Sie TortoiseSVN → Patch erzeugen...

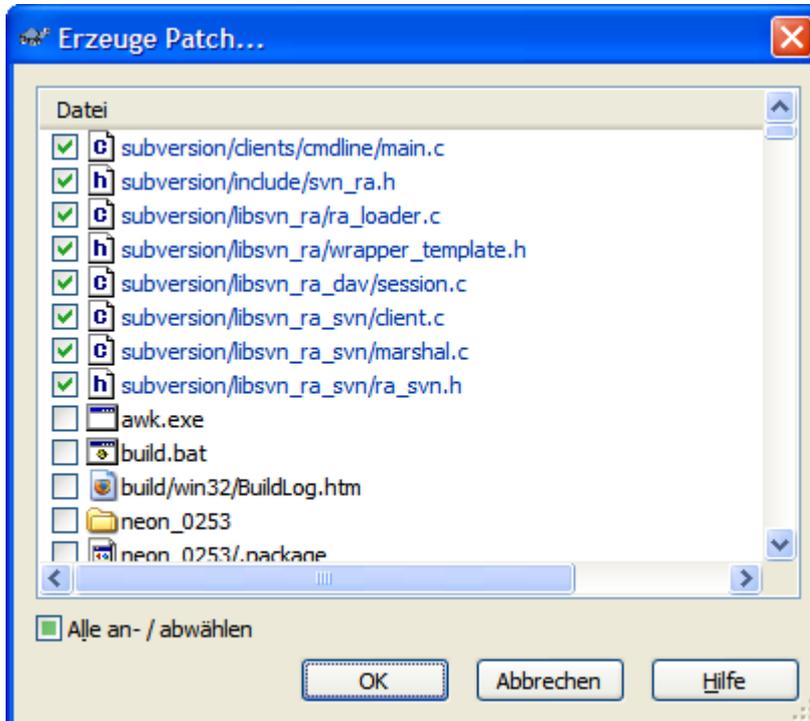


Abbildung 4.43. Der „Erzeuge Patch“-Dialog.

Sie können nun die Dateien auswählen, welche Sie in dem Patch drin haben möchten. Genauso wie wenn sie die Dateien übertragen würden. Dies wird eine Datei erstellen mit all den Änderungen seit der letzten Übertragung der ausgewählten Dateien.

Die Spalten in diesem Dialog können, genau wie im Auf Änderungen prüfen-Dialog, angepasst werden. Lesen Sie in [Abschnitt 4.7.3](#), „Prüfe auf Änderungen“ nach, wie das geht.

Sie können separate Patches erzeugen für die verschiedenen Änderungen an den Dateien. Wenn Sie jedoch eine Patchdatei erstellen, dann weitere Änderungen an *denselben* Dateien vornehmen und eine weitere Patchdatei erstellen, wird die zweite Patchdatei *beide* Sätze von Änderungen enthalten.

Speichern Sie die Datei unter einem Namen Ihrer Wahl. Patchdateien können jede beliebige Dateierstreckung haben. Eine Konvention ist jedoch, `.patch` dafür zu verwenden. Nun sind Sie so weit. Sie können Ihre Patchdatei an die Entwickler schicken.

Sie können den Patch auch in der Zwischenablage anstatt einer Datei speichern, zum Beispiel weil Sie den Patch per E-Mail zur Begutachtung weiterschicken wollen. Auch wenn sie zwei Arbeitskopien auf einem Rechner haben und gezielt Änderungen von einer in die andere übertragen wollen, ist ein Patch in der Zwischenablage der bequemste Weg.

4.22.2. Eine Patchdatei anwenden

Patchdateien werden stets innerhalb Ihrer Arbeitskopie angewendet. Dies muss innerhalb desselben Ordners durchgeführt werden, in dem die Patchdatei erstellt wurde. Wenn Sie nicht genau wissen, welcher Ordner das war, schauen Sie in die erste Zeile der Patchdatei. Wenn z.B. die erste bearbeitete Datei `doc/source/deutsch/kapitel1.xml` hieß und die erste Zeile in der Patchdatei `Index: deutsch/kapitel1.xml` lautet, dann müssen sie den Patch auf den `doc/source/` Ordner anwenden. Falls Sie den Patch in der richtigen Arbeitskopie aber nicht im richtigen Ordner anwenden, wird TortoiseSVN das bemerken und den Pfad vorschlagen, den es für richtig hält.

Um den Patch anwenden zu können, benötigen Sie zumindest Lesezugriff auf das Projektarchiv, denn das Programm zum Zusammenführen (Patching) der Daten muss eventuell auf dieses zurückgreifen, um die Änderungen auf die Revision im Projektarchiv zurückzuführen, auf der sie basieren.

Vom Kontextmenü des betroffenen Ordners aus wählen Sie TortoiseSVN → Patch anwenden... Dies öffnet einen Dateidialog von TortoiseMerge, in dem sie die anzuwendende Patchdatei auswählen können. Standardmäßig werden nur `.patch` Dateien angezeigt, aber sie können auch „Alle Dateien“ auswählen. Wenn Sie einen Patch in der Zwischenablage gespeichert haben, können Sie Aus Zwischenablage öffnen im Datei Öffnen-Dialog aufrufen.

Falls die Patchdatei einen `.patch` oder `.diff` Anhang hat, können Sie mit einem Rechtsklick auf die Datei direkt TortoiseSVN → Patch anwenden... wählen. In diesem Fall werden Sie gebeten, eine Arbeitskopie anzugeben.

Dies sind zwei verschiedene Wege, um dasselbe Ziel zu erreichen. Mit der ersten Methode geben Sie die Arbeitskopie vor und suchen die Patchdatei. Mit der zweiten Methode geben Sie die Patchdatei vor und suchen die Arbeitskopie.

Sobald Sie eine Datei gewählt haben, wird TortoiseMerge gestartet, um die Änderungen in Ihrer Arbeitskopie zusammenzuführen. Ein kleines Fenster zeigt die Liste der Dateien an, die durch die Patchdatei geändert würden. Führen Sie einen Doppelklick auf die einzelnen Dateien aus und die Änderungen aus der Patchdatei werden mit Ihrer Arbeitskopie zusammengeführt. Prüfen Sie die Änderungen und speichern Sie die geänderten Dateien.

Die Patchdatei wurde nun in Ihrer Arbeitskopie angewendet, so dass die Veränderungen nun zum Projektarchiv übertragen werden müssen, um anderen Projektteilnehmern zur Verfügung zu stehen.

4.23. Wer hat welche Zeile geändert?

Manchmal möchten Sie nicht nur erfahren, welche Zeilen sich zwischen verschiedenen Revisionen geändert haben, sondern auch, wer genau für welche Zeile in einer Datei verantwortlich ist. Dazu gibt es den Befehl TortoiseSVN → Annotieren....

Dieser Befehl listet für jede Zeile einer Datei den Autor und die Revision der letzten Änderung dieser Zeile auf.

4.23.1. Annotieren für Dateien

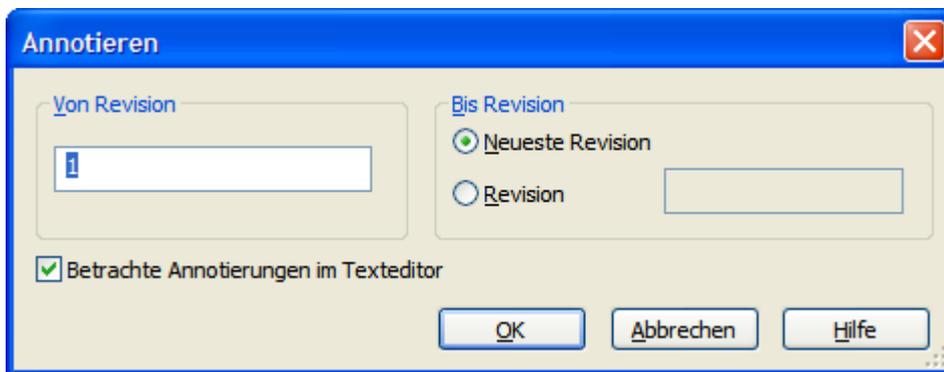


Abbildung 4.44. Der Annotieren-Dialog

Wenn Sie nur an den Änderungen in neueren Revisionen interessiert sind, können Sie die Revision festlegen, bei der das Annotieren beginnen soll. Wenn Sie *jede* Revision annotieren wollen, setzen Sie diesen Wert auf 1.

Standardmäßig wird die annotierte Datei in *TortoiseBlame* angezeigt, welches die verschiedenen Revisionen hervorhebt. Wenn Sie die annotierte Datei bearbeiten oder ausdrucken möchten, aktivieren Sie *Betrachte Annotierungen im Texteditor*.

Sie können festlegen, wie Änderungen an Zeilenumbrüchen und Leerzeichen behandelt werden sollen. Diese Optionen sind in [Abschnitt 4.10.2, „Zeilende- und Leerzeichenoptionen“](#) beschrieben. Die Vorgabe ist, dass alle Zeilenenden- und Leerzeichenänderungen als echte Änderungen behandelt werden, aber wenn Sie solche Änderungen ignorieren und den Originalautor finden wollen, können Sie hier die entsprechende Option wählen.

Sobald Sie auf OK klicken, beginnt TortoiseSVN mit der Analyse der Daten dieser Datei. Beachten Sie bitte, dass dies mehrere Minuten dauern kann, je nachdem wie schnell Ihre Verbindung zum Projektarchiv ist. Sobald der TortoiseSVN mit der Analyse fertig ist wird das Ergebnis der Analyse in *TortoiseBlame* angezeigt.

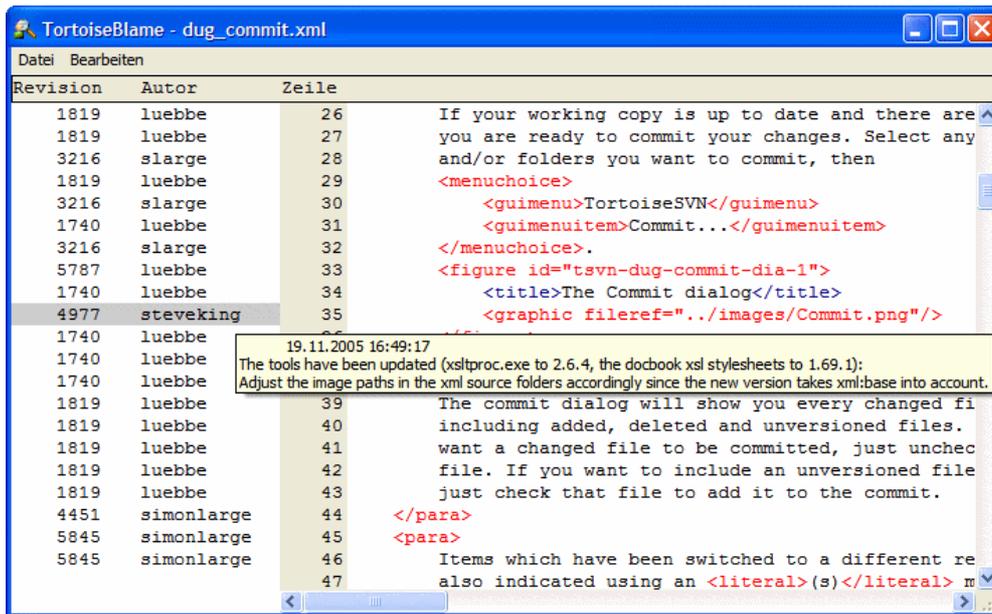


Abbildung 4.45. TortoiseBlame

TortoiseBlame, das zusammen mit TortoiseSVN installiert wird, macht eine solche Annotierung leichter lesbar. Wenn Sie mit dem Mauszeiger über eine Zeile in der linken Spalte fahren, werden alle Zeilen, die in derselben Revision geändert wurden mit einer dunkleren Farbe dargestellt. Zeilen von anderen Revisionen, aber demselben Autor werden mit einem helleren Hintergrund dargestellt. Die Einfärbung funktioniert eventuell nicht optimal wenn Sie Ihren Monitor auf nur 256 Farben eingestellt haben.

Wenn Sie auf eine Zeile Links klicken werden alle Zeilen derselben Revision markiert, und Zeilen von anderen Revisionen aber demselben Autor werden in einer anderen Farbe dargestellt. Diese Markierung bleibt bestehen, so dass Sie den Mauszeiger bewegen können ohne die Einfärbung zu verlieren. Klicken Sie nochmals auf diese Revision um die Markierung wieder aufzuheben.

Die Revisionskommentare (Logmeldungen) werden in einem Hinweistext angezeigt, wenn sich die Maus über der Informationsspalte befindet. Falls sie diese Logmeldung in die Zwischenablage kopieren wollen, benutzen Sie dazu das Kontextmenü, das bei einem Rechtsklick auf die Informationsspalte erscheint.

Sie können innerhalb der Annotierungen suchen, indem Sie **Bearbeiten** → **Suchen...** aufrufen. Diese Funktion erlaubt Ihnen, nach Revisionsnummern, Autoren und dem Inhalt der Datei selbst zu suchen. Logmeldungen werden nicht durchsucht. Für diese Funktion steht der Log-Dialog zur Verfügung.

Sie können auch mittels **Bearbeiten** → **Gehe zu Zeile...** in eine bestimmte Zeile der Datei springen.

Wenn die Maus sich über dem Annotieren Fenster befindet, steht ein Kontextmenü zur Verfügung, mit dessen Hilfe Sie Revisionen vergleichen und die Historie, unter Verwendung der Revisionsnummer auf die die Maus zeigt, betrachten können. Kontextmenü → **Vorherige Revision annotieren** erzeugt eine Annotierung derselben Datei unter Verwendung der vorherigen Revisionsnummer als Obergrenze. Dies zeigt Ihnen den Zustand der Datei bevor die Zeile, die Sie gerade betrachten, geändert wurde, an. Kontextmenü **Zeige Änderungen**

Wenn Sie einen schnell sehen möchten, wo die ältesten und die neuesten Änderungen sind, wählen Sie **Ansicht** → **Farbcodierung für Alter der Zeilen**. Dann werden neue Zeilen in rot, ältere Zeilen in blau mit einem Farbgradienten hinterlegt. Die Standardfarben sind recht hell, können aber in den TortoiseBlame Einstellungen geändert werden.

Wenn Sie die Protokollierung der Datenintegration verwenden, wird TortoiseBlame Zeilen, die als Folge des Zusammenführens geändert wurden, mit dem Autor und der Revision der letzten Änderung

der Originaldatei anzeigen und nicht mit der Revision, in der die Daten zusammengeführt wurden. In diesen Zeilen werden Autor und Revision in Kursivschrift dargestellt. Wenn Sie nicht möchten, dass die zusammengeführten Zeilen so dargestellt werden, deaktivieren Sie die Option Zusammenführen-Info einschließen.

Wenn Sie sehen wollen, welche Pfade zusammengeführt wurden, wählen Sie Ansicht → Zusammengeführte Pfade.

Die Einstellungen für TortoiseBlame können Sie über TortoiseSVN → Einstellungen auf dem Karteireiter TortoiseBlame anpassen. Siehe [Abschnitt 4.30.9](#), „TortoiseBlame Einstellungen“.

4.23.2. Unterschiede annotieren

Eine Einschränkung des Annotierens ist, dass die Datei nur in einer bestimmten Revision sowie die letzte Person, die eine bestimmte Zeile geändert hat, angezeigt wird. Manchmal möchte man zusätzlich auch wissen, welche Änderungen durchgeführt wurden. Dazu bietet TortoiseSVN eine Kombination aus Vergleichen und Annotieren.

Der Log-Dialog bietet Ihnen mehrere Möglichkeiten dazu.

Revisionen annotieren

Selektieren Sie im oberen Teil zwei Revisionen und wählen dann Kontextmenü → Revisionen annotieren. Dadurch wird die Annotierungsinformation für die zwei Revisionen geholt und anschließend das Vergleichsprogramm gestartet, um die zwei Annotierungen gegenüberzustellen

Änderungen annotieren

Markieren Sie eine Revision in der oberen Liste und eine Datei aus der unteren Liste. Wählen Sie dann Kontextmenü → Unterschiede annotieren. Dies wird die Annotierungen für die gewählte sowie die vorherige Revision erstellen und im Vergleichsprogramm anzeigen.

Vergleiche und Annotiere mit Arbeitskopie BASE

Zeigen Sie das Log für eine einzelne Datei an und markieren Sie in der oberen Liste eine einzelne Revision. Wählen Sie Kontextmenü → Vergleiche und Annotiere mit Arbeitskopie BASE. Dies wird die Annotierungen für die gewählte Revision sowie für die BASE Revision der Arbeitskopie erstellen und im Vergleichsprogramm anzeigen.

4.24. Projektarchivbetrachter

Vielleicht möchten Sie manchmal direkt im Projektarchiv Änderungen vornehmen, ohne eine Arbeitskopie zu haben oder gar auschecken zu müssen. In solchen Situationen kommt der *Projektarchivbetrachter* zum Einsatz. Was der Explorer und die überlagerten Symbole für Ihre Arbeitskopie sind, ist der Projektarchivbetrachter für das Projektarchiv.

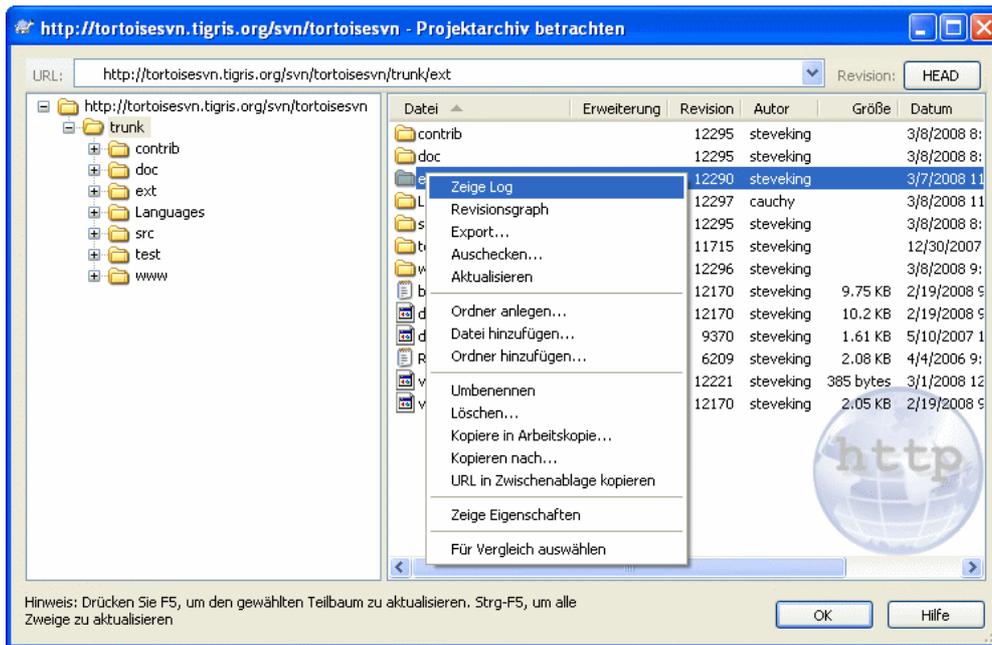


Abbildung 4.46. Projektarchivbetrachter

Mit dem Projektarchivbetrachter können Sie Befehle wie kopieren, löschen, umbenennen, ... direkt im Projektarchiv ausführen.

Der Projektarchivbetrachter sieht dem Windows Explorer sehr ähnlich. Im Gegensatz zu diesem zeigt er den Inhalt eines Projektarchivs anstelle von lokalen Dateien an. Auf der linken Seite sehen Sie einen Verzeichnisbaum und auf der rechten Seite den Inhalt des gewählten Verzeichnisses. In der Adresszeile können Sie die URL des Projektarchivs sowie die Revision, die Sie betrachten wollen, angeben.

Wie im Windows Explorer können Sie auf die Spaltenköpfe klicken, um den Inhalt zu sortieren. In jedem der beiden Bereiche stehen Ihnen Kontextmenüs zur Verfügung.

Das Kontextmenü einer Datei erlaubt Ihnen:

- Die gewählte Datei entweder mit dem Standardprogramm für den Dateityp oder mit einem von Ihnen gewählten Programm öffnen.
- Eine nicht versionierte Kopie der Datei auf Ihrer Festplatte speichern.
- Zeigt das Revisionslog der Datei oder den Graphen aller Revisionen, so dass Sie sehen können, woher die Datei stammt.
- Annotiert jede Zeile der Datei mit dem Autor und dem Änderungsdatum
- Datei löschen oder umbenennen
- Erstellt eine Kopie der Datei, entweder in einem anderen Teil des Projektarchivs oder in einer Arbeitskopie die auf dem selben Projektarchiv basiert.
- Die Dateieigenschaften anzeigen/bearbeiten.

Das Kontextmenü eines Ordners erlaubt Ihnen:

- Zeigt das Revisionslog des Ordners oder den Graphen aller Revisionen, so dass Sie sehen können, woher der Ordner stammt.
- Exportiert den Ordner in eine lokale, unversionierte Kopie.

- Check den Ordner in eine lokale Arbeitskopie aus.
- Einen neuen Ordner im Projektarchiv anlegen
- Dateien oder Ordner zum Projektarchiv hinzufügen
- Ordner löschen oder umbenennen
- Erstellt eine Kopie des Ordners, entweder in einem anderen Teil des Projektarchivs oder in einer Arbeitskopie die auf dem selben Projektarchiv basiert.
- Ordneigenschaften betrachten/ändern.
- Markiert den Ordner für den Vergleich. Ein markierte Ordner wird durch Fettdruck gekennzeichnet
- Den Ordner mit einem bereits gewählten Ordner vergleichen. Entweder können Sie den Vergleich als Standard-Diff oder als Liste von Dateien anzeigen lassen, die Sie dann einzeln miteinander vergleichen. Diese Funktion ist sehr nützlich, wenn Sie z.B. herausfinden wollen, was sich zwischen zwei Marken geändert hat.

Wenn Sie im rechten Bereich zwei Ordner wählen, können Sie sich die Unterschiede entweder als Standard-Diff oder als Liste von Dateien anzeigen lassen, die Sie dann einzeln miteinander vergleichen.

Wenn Sie mehrere Ordner im rechten Bereich wählen, können Sie diese Ordner alle gleichzeitig in einen gemeinsamen Elternordner auschecken.

Wenn Sie zwei Marken wählen, die von derselben Basis erstellt wurden (typischerweise `/trunk/`), können Sie **Kontext Menü** → **Zeige Log...** verwenden, um sich die Unterschiede zwischen den beiden Marken anzeigen zu lassen.

Sie können mit **F5** die Anzeige aktualisieren. Wenn Sie die Daten von Knoten, die bisher noch nicht geöffnet wurden, laden wollen, verwenden Sie dafür **Strg-F5**. Danach erfolgt das Öffnen eines Knotens ohne Verzögerung, da keine Informationen mehr über das Netzwerk geladen werden müssen.

Sie können den Projektarchivbetrachter auch für Ziehen und Ablegen Aktionen verwenden. Wenn Sie eine Datei aus dem Explorer in den Projektarchivbetrachter ziehen, wird sie in das Projektarchiv importiert. Wenn Sie auf diese Weise mehrere Objekte gleichzeitig importieren, wird jedes Objekt in einer eigenen Revision importiert.

Wenn Sie ein Objekt innerhalb des Projektarchivs verschieben wollen, ziehen Sie es einfach mit der linken Maustaste an seinen neuen Ort. Wenn Sie stattdessen eine Kopie erstellen wollen, halten Sie dabei die **Strg** Taste gedrückt. Beim Kopieren erhält der Mauszeiger, wie im Explorer, ein „Plus“ Symbol.

Wenn Sie eine Datei oder einen Ordner an einen anderen Platz kopieren/verschieben und gleichzeitig einen neuen Namen vergeben möchten, können Sie die Datei oder den Ordner mit Rechts-Ziehen oder **Strg-Rechts-Ziehen** anstatt Links-Ziehen verschieben. In diesem Falle wird ein Umbenennen-Dialog angezeigt, in dem Sie den neuen Namen für die Datei oder den Ordner eingeben können.

Immer, wenn Sie auf diese Art Änderungen im Projektarchiv vornehmen, wird ein Logdialog angezeigt, in den Sie eine Meldung eingeben können. Wenn Sie versehentlich etwas verschoben haben, können Sie damit die Aktion abbrechen.

Manchmal werden Sie, wenn Sie einen Pfad öffnen wollen, eine Fehlermeldung anstelle der Details erhalten. Das kann daran liegen, dass Sie eine falsche URL angegeben haben, nicht über entsprechende Zugriffsrechte verfügen oder dass ein Serverfehler vorliegt. Wenn Sie diese Meldung z.B. in eine E-Mail kopieren wollen, machen Sie einen Rechtsklick darauf und wählen Sie **Kontextmenü** → **Fehlermeldung** in die Zwischenablage kopieren oder einfach **Strg+C**.

4.25. Revisionsgraphen

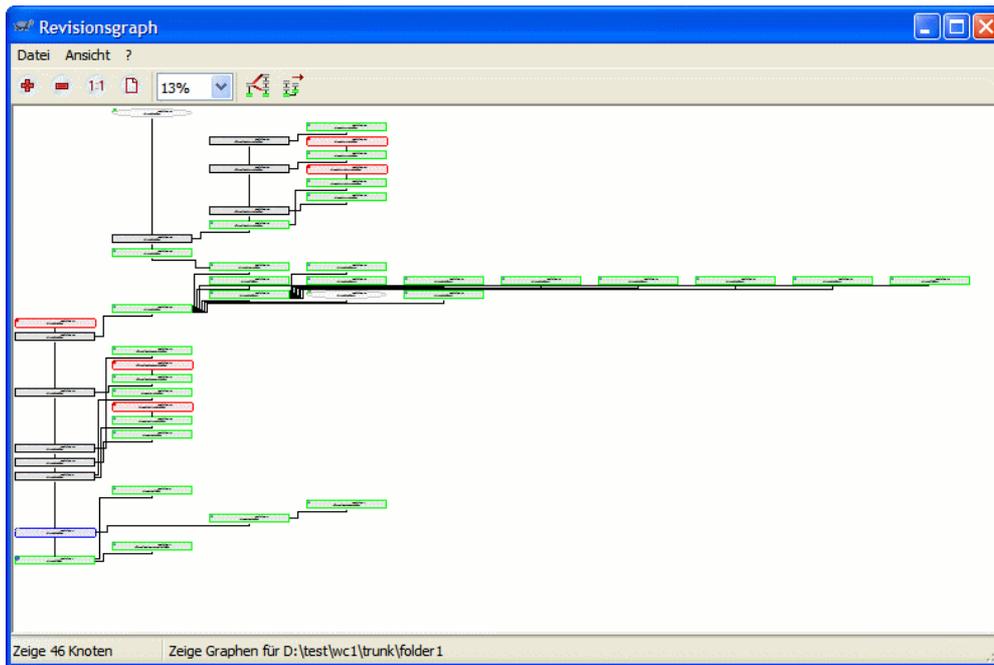


Abbildung 4.47. Ein Revisionsgraph

Manchmal möchten Sie wissen, wann Zweige oder Marken vom Stamm abgespalten wurden. Der ideale Weg, um dies zu untersuchen, ist ein Graph oder eine Baumstruktur. Dafür gibt es den TortoiseSVN → Revisionsgraph

Dieser Befehl analysiert die Revisionsgeschichte und versucht eine Baumstruktur mit den Zeitpunkten darzustellen, in denen Kopien oder Zweige/Marken erstellt bzw. gelöscht wurden.



Wichtig

Um den Revisionsgraphen zu erzeugen, muss TortoiseSVN alle Logmeldungen von der Basis des Projektarchivs holen. Überflüssig zu sagen, dass diese Aktion in Abhängigkeit von Netzwerk- und Servergeschwindigkeit bereits bei wenigen tausend Revisionen mehrere Minuten in Anspruch nehmen kann. Wenn Sie sich den Revisionsgraphen des *Apache* Projektes mit derzeit über 500.000 Revisionen anzeigen lassen, sollten Sie sich auf eine längere Wartezeit einstellen.

Die gute Nachricht ist, dass Sie, wenn Sie den Log-Puffer benutzen, diese Wartezeit nur einmal ertragen müssen. Danach werden die Logdaten lokal vorgehalten. Der Log-Puffer wird in den TortoiseSVN Einstellungen aktiviert.

4.25.1. Knoten des Revisionsgraphen

Jeder Knoten des Revisionsgraphen entspricht einer Revision im Projektarchiv in der sich etwas in dem Baum, den Sie gerade betrachten, geändert hat. Verschiedene Knotentypen können anhand ihrer Form und Farbe unterschieden werden. Die Formen sind fest vorgegeben, die Farben können über TortoiseSVN → Einstellungen geändert werden.

Hinzugefügte oder kopierte Objekte

Elemente, die hinzugefügt oder durch Kopieren erstellt wurden, werden als abgerundetes Rechteck dargestellt. Die Standardfarbe ist grün. Marken und der Stamm werden als Sonderfall behandelt und in eine anderen Schattierung dargestellt, die Sie in TortoiseSVN → Optionen anpassen können

Gelöschte Objekte

Gelöschte Elemente wie z.B. ein Zweig der nicht länger benötigt wird, werden als Achteck dargestellt. Die Standardfarbe ist rot.

Umbenannte Objekte

Umbenannte Objekte werden ebenfalls als Achteck angezeigt, aber die Standardfarbe ist blau.

Spitze eines Zweiges

Der Graph beschränkt sich normalerweise darauf, die Verzweigungspunkte anzuzeigen. Manchmal ist es aber auch nützlich, die entsprechenden HEAD Revisionen jedes Zweiges zu sehen. Wenn Sie **Zeige HEAD Revisionen** aktivieren, wird jede HEAD Revision als Ellipse angezeigt. Beachten Sie bitte, dass HEAD sich in diesem Fall auf die zuletzt übertragene Revision des Zweiges und nicht des gesamten Projektarchivs bezieht.

Revision der Arbeitskopie

Falls Sie den Revisionsgraphen aus einer Arbeitskopie heraus aufgerufen haben, können Sie sich die BASE Revision der Arbeitskopie durch eine fette Umrandung anzeigen lassen, indem Sie **Zeige Revision der Arbeitskopie** aktivieren.

Veränderte Arbeitskopie

Falls Sie den Revisionsgraphen aus einer Arbeitskopie heraus aufgerufen haben, können sie sich den Änderungsstatus Ihrer Arbeitskopie mit **Änderungen in Arbeitskopie** anzeigen in einem zusätzlichen Knoten anzeigen lassen. Diese Knoten ist standardmäßig rot mit fettem Rand.

Normale Objekte

Alle anderen Elemente werden durch ein einfaches Rechteck dargestellt.

Beachten Sie, dass in der Standardansicht der Graph nur die Punkte zeigt, an denen Elemente hinzugefügt, kopiert oder gelöscht wurden. Alle Revisionen eines Projektes anzuzeigen würde in nicht-trivialen Fällen einen riesigen Graphen erzeugen. Wenn Sie wirklich *alle* Revisionen sehen wollen, können Sie dies über einen Schalter in der Werkzeuleiste oder im Menü **Ansicht** aktivieren.

Die Standardansicht (Gruppierung aus) platziert die Knoten so, dass die vertikale Position strikt der Revisionsreihenfolge entspricht, so dass Sie einen Hinweis auf die Abfolge der Ereignisse haben. Befinden sich zwei Knoten in derselben Spalte, ist die Reihenfolge offensichtlich. Bei benachbarten Spalten ist der Versatz viel kleiner und die Reihenfolge weniger offensichtlich, da nicht darauf geachtet werden muss, dass die Knoten sich nicht überlappen. Solche Optimierungen sind nötig, um komplexe Graphen auf eine vertretbare Größe zu reduzieren. Bitte beachten Sie, dass diese Ordnung die *Kante* des Knotens auf der *älteren* Seite als Referenz verwendet, d.h. die untere Kante des Knotens, wenn der Graph mit dem ältesten Knoten unten angezeigt wird. Die Referenzkante ist wichtig, weil die Formen der Knoten nicht alle dieselbe Höhe haben.

4.25.2. Die Ansicht ändern

Da ein Revisionsgraph häufig sehr komplex ist, stehen einige Funktionen zur Verfügung, mit denen Sie die Ansicht an Ihre Wünsche anpassen können. Diese stehen im Menü **Ansicht** sowie in der Menüleiste zur Verfügung.

Zweige zusammenfassen

Die Vorgabe (Gruppieren aus) sortiert alle Zeilen strikt nach der Revisionsnummer. Dadurch belegen langlebige Zweige selbst mit nur wenigen Änderungen eine ganze Spalte und der Graph wird sehr breit.

Dieser Modus gruppiert Änderungen in Zweigen, so dass es keine globale Revisionsordnung gibt. Aufeinanderfolgende Revisionen eines Zweiges werden (häufig) in aufeinanderfolgenden Zeilen angezeigt. Unterzweige hingegen werden so angeordnet, dass spätere Zweige in derselben Spalte über früheren liegen, um den Graphen schmal zu halten. Als Ergebnis kann eine Zeile Änderungen aus verschiedenen Revisionen enthalten.

Älteste nach oben

Normalerweise zeigt der Graph die ältesten Revisionen unten und der Revisionsbaum wächst nach oben. Mit dieser Option lassen Sie den Graphen von oben nach unten wachsen.

Bäume oben ausrichten

Wenn ein Graph in mehrere kleine Bäume aufgespalten wird, werden diese, je nachdem ob Sie **Zweige zusammenfassen** aktiviert haben, in der natürlichen Revisionsreihenfolge oder an der Unterkante des Fensters angezeigt. Mit diese Option werden die Bäume von der Oberkante aus aufgebaut.

Überschneidungen reduzieren

Wenn der Graph von vielen Linien durchschnitten wird, können Sie die Ansicht mit dieser Funktion aufräumen. Dadurch werden die Spalten unter Umständen an weniger logischen Stellen angezeigt und es dauert unter Umständen länger, bis der Graph gezeichnet ist.

Differenzielle Pfadnamen

Lange Pfadnamen verbrauchen viel Platz und machen die Knoten unnötig breit. Mit dieser Option werden nur die unterschiedlichen Teile des Pfades angezeigt und die übereinstimmenden Teile durch Punkte ersetzt. Wenn Sie z.B. `/branches/1.2.x/doc/html` aus `/trunk/doc/html` erstellen, kann der Zweig in der kompakten Form als `/branches/1.2.x/. .` angezeigt werden, weil die zwei Ebenen `doc` und `html` sich nicht geändert haben.

Alle Revisionen zeigen

Diese Funktion zeigt jede Revision des Graphen, in der sich etwas geändert hat, als separaten Knoten an. Bei älteren Projekten mit vielen Revisionsnummern kann das zu einem riesigen Graphen führen.

HEAD Revisionen zeigen

Diese Option stellt sicher, dass immer die neuesten Revisionen aller Zweige im Graphen angezeigt werden.

Exakte Quellen der Kopien

Wenn eine Verzweigung/Markierung angelegt wird, ist das Standardverhalten, den Zweig so anzuzeigen, als sei er vom letzten geänderten Knoten abgeleitet. Genau genommen ist das inkorrekt, da Zweige häufig von der HEAD Revision und nicht einer bestimmten Revision erstellt werden. Hiermit ist es möglich, die korrektere (aber weniger nützliche) Revision anzuzeigen, aus der der Zweig wirklich angelegt wurde. Beachten Sie bitte, dass diese Revision jünger als die HEAD Revision des Quellzweiges sein kann.

Zweige ausblenden

Wenn ein Projekt viele Marken enthält, würde es viel Platz beanspruchen, alle diese Marken darzustellen und wichtigere Teile der Verzweigungsstruktur verschleiern. Gleichzeitig möchten Sie einfach auf die Marken zugreifen können, um Revisionen zu vergleichen. Diese Option verbirgt die Knoten für Marken und zeigt sie stattdessen im Hinweistext des Knotens an, von dem aus sie erstellt wurden. Ein Symbol am rechten Rand des Quellknotens zeigt an, dass eine Marke erstellt wurde.

Gelöschte Pfade verbergen

Blendet Pfade aus, die in der HEAD Revision des Projektarchivs nicht mehr existieren, z.B. gelöschte Pfade.

Ungenutzte Zweige verbergen

Blendet Pfade aus in denen keine Änderungen an dem gewählten Objekt in das Projektarchiv übertragen wurden. Das bedeutet nicht, dass der Zweig nicht benutzt wurde, sondern dass *dieser* Teil des Zweiges nicht verändert wurde.

Arbeitskopie Revisionen zeigen

Hebt die Revision im Graphen hervor, die der Revison des Objekts entspricht, für das Sie den Graphen anzeigen. Wenn Sie gerade Ihre Arbeitskopie aktualisiert haben wird das HEAD sein, wenn aber andere nach der Aktualisierung noch Änderungen übertragen haben, kann Ihre Arbeitskopie ein paar Revisionen niedriger sein. Der Knoten wird durch eine fette Umrandung gekennzeichnet.

Änderungen der Arbeitskopie zeigen

Falls Ihre Arbeitskopie lokale Änderungen enthält, wird sie durch diese Option als ein zusätzlicher, elliptischer Knoten angezeigt, der mit der Revision zu der Ihre Arbeitskopie aktualisiert wurde, verknüpft ist. Sie müssen den Graphen unter Umständen mit **F5** aktualisieren, um neuere Änderungen angezeigt zu bekommen.

Filter

Manchmal zeigt der Revisionsgraph mehr Details an als Sie eigentlich sehen wollen. Diese Funktion öffnet einen Dialog, mit dem Sie den angezeigten Revisionsbereich einschränken oder bestimmte Pfade über ihre Namen ausschließen können.

Baumstreifen

Falls der Graph mehrere Bäume enthält, ist es oft nützlich, zur Unterscheidung der Bäume abwechselnde Farben im Hintergrund zu haben.

Gesamtübersicht anzeigen

Zeigt ein kleines Übersichtsbild des gesamten Graphen mit dem aktuellen Ausschnitt als Rechteck. Sie können dieses Rechteck verschieben, um einfach durch den Graphen zu navigieren. Beachten Sie bitte, dass die Übersicht bei sehr großen Graphen aufgrund der starken Verkleinerung nutzlos ist und in diesen Fällen nicht angezeigt wird.

4.25.3. Den Graphen verwenden

Um die Navigation durch einen großen Graphen zu vereinfachen, steht das Übersichtsfenster zur Verfügung. Darin wird der vollständige Graph sowie der aktuelle Ausschnitt hervorgehoben dargestellt. Sie können den aktuellen Ausschnitt verschieben, um den angezeigten Bereich zu verändern.

Das Datum, der Autor und die Logmeldung der Revision werden in einem Hinweistext angezeigt, sobald die Maus über ein Element bewegt wird.

Wenn Sie mit **Strg**-Linksklick zwei Revisionen markieren, können Sie sich über das Kontextmenü die Unterschiede zwischen diesen Revisionen anzeigen lassen. Sie können wählen, ob Sie die Unterschiede am Anfang der Zweige oder die Unterschiede an den Enden, sprich den HEAD Revisionen vergleichen wollen.

Sie können sich die Unterschiede im Standard-Diff Format anzeigen lassen, das alle Unterschiede in einer einzelnen Datei mit minimalem Kontext zeigt. Wenn Sie Kontextmenü → Revisionen vergleichen wählen, wird Ihnen eine Liste der geänderten Dateien angezeigt. Mit einem Doppelklick auf einen Dateinamen werden beide Revisionen geholt und in Ihrem Diff Betrachter angezeigt.

Mit einem Rechtsklick auf eine Revision, können Sie sich per Kontextmenü → Zeige Log die Historie anzeigen lassen.

Sie können auch die Änderungen der gewählten Revisionen in einer anderen Arbeitskopie zusammenführen. Ein Ordnerauswahldialog ermöglicht es Ihnen, eine andere Arbeitskopie als Ziel zu wählen, aber danach gibt es weder einen Bestätigungsdialog noch eine Möglichkeit für einen Testlauf. Es ist eine gute Idee, Änderungen in einer unmodifizierten Arbeitskopie zusammenzuführen, so dass Sie diese stets rückgängig machen können, wenn etwas nicht klappt! Dies ist eine nützliche Funktion, wenn Sie bestimmte Revisionen von einem Zweig in einen anderen übertragen wollen.



Lernen Sie, den Revisionsgraphen zu lesen

Neueinsteiger stehen unter Umständen vor dem Problem, dass der Graph nicht das anzeigt, was sie erwarten. Falls zum Beispiel eine Revision mehrere Kopien oder Zweige einer Datei oder eines Ordners ändert, wird es mehrere Knoten für diese einzelne Revision geben. Es ist ratsam, mit der linken Option in der Werkzeugleiste zu beginnen und den Graphen schrittweise an die eigene Vorstellung anzupassen.

Sämtliche Filteroptionen versuchen so viel Information wie möglich beizubehalten. Das kann unter Anderem dazu führen, dass sich die Farbe einiger Knoten ändert. Wann immer ein Ergebnis unerwartet ist, nehmen Sie die letzte Filteroperation zurück und versuchen Sie, herauszufinden, was das Besondere an dieser Revision oder diesem Zweig ist. In den meisten Fällen ist das zunächst erwartete Filterergebnis ungenau oder irreführend.

4.25.4. Die Ansicht aktualisieren

Wenn Sie den Server auf neue Informationen überprüfen wollen, können Sie die Sicht einfach per **F5** aktualisieren. Falls Sie den Log-Puffer verwenden (standardmäßig aktiv), wird das Projektarchiv auf neue Übertragungen abgefragt und nur diese werden geholt. Falls der Log-Puffer offline war, wird gleichzeitig versucht, ihn wieder online zu schalten.

Falls Sie den Log-Puffer verwenden und denken, dass sich der Inhalt oder der Autor einer Logmeldung geändert haben, sollten Sie den Log-Dialog aufrufen, um darüber die gewünschten Meldungen zu aktualisieren. Da der Revisionsgraph von der Wurzel des Projektarchivs ausgeht, müssten wir den gesamten Log-Puffer ungültig machen und erneut aufbauen, was *sehr viel* Zeit in Anspruch nehmen kann.

4.25.5. Zweige ausdünnen

Es kann schwierig sein, in einem großen Baum zu navigieren und eventuell möchten Sie Teile ausblenden oder den Graphen in einen Wald aus kleineren Bäumen aufspalten. Wenn Sie mit der Maus über den Punkt fahren, an dem ein Verweis einen Knoten betritt oder verlässt, werden Ihnen eine oder mehrere Schaltflächen mit den entsprechenden Funktionen angezeigt.



Klicken Sie auf die Minus-Schaltfläche, um den angehängten Teilbaum zusammenzuklappen.



Klicken Sie auf die Plus-Schaltfläche, um den Baum aufzuklappen. Wenn ein Baum zusammengeklappt wird, bleibt diese Schaltfläche sichtbar, um den verborgenen Teilbaum anzuzeigen.



Klicken Sie auf die Kreuz-Schaltfläche, um den angehängten Teilbaum abzuspalten und als separaten Baum im Graphen anzuzeigen.



Klicken Sie auf die Kreis-Schaltfläche, um einen abgespaltenen Teilbaum wieder einzuhängen. Falls ein Teilbaum abgespalten wurde, bleibt diese Schaltfläche sichtbar, um anzuzeigen, dass es einen separaten Teilbaum gibt.

Klicken Sie auf den Hintergrund des Graphen, um das Kontextmenü angezeigt zu bekommen. Es stellt Ihnen die Funktionen **Alle aufklappen** und **Alle verbinden**. Falls noch kein Ast zugeklappt oder aufgetrennt wurde, wird das Kontextmenü nicht angezeigt.

4.26. Eine Arbeitskopie exportieren

Es kommt vor dass Sie eine Kopie Ihres Projektes möchten, aber ohne die administrativen Ordner (`.svn`), zum Beispiel weil Sie das Projekt zippen möchten. Statt dass Sie nun eine Kopie von Ihrer Arbeitskopie erstellen und dann von Hand alle `.svn` Ordner löschen können Sie den Befehl `TortoiseSVN →`

Exportieren... verwenden. Exportieren von einer URL und von einer Arbeitskopie werden leicht unterschiedlich behandelt.

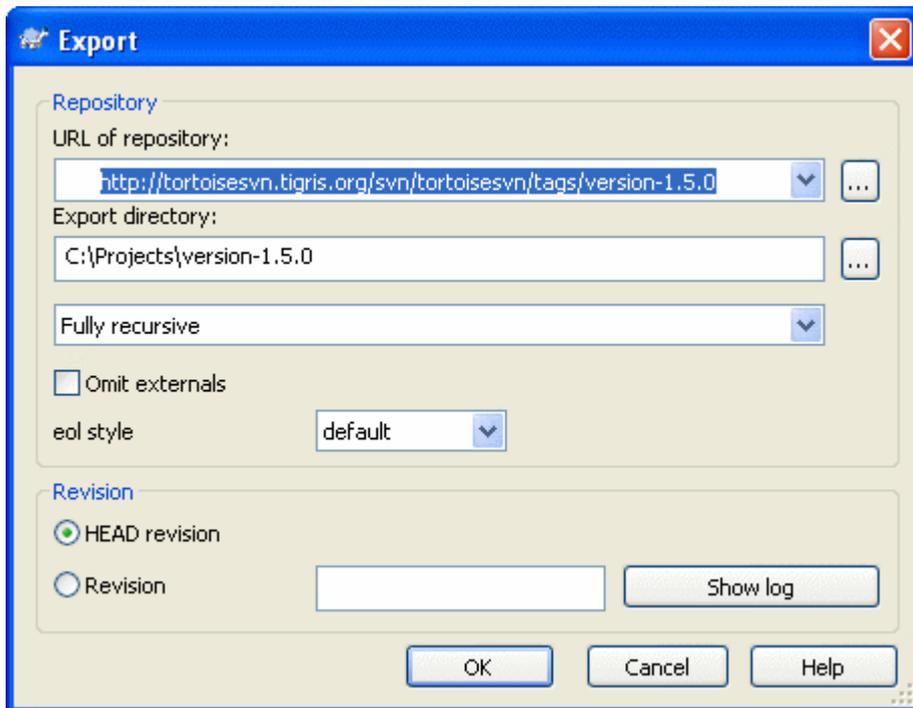


Abbildung 4.48. Exportiere von URL

Wenn Sie diesen Befehl auf einem nicht versionierten Ordner ausführen, wird TortoiseSVN annehmen, dass es sich dabei um das Ziel des Exports handelt und einen Dialog anzeigen, in dem Sie die URL von der exportiert werden soll, angeben können. Dieser Dialog bietet die Optionen nur den obersten Ordner zu exportieren, externe Verweise auszulassen und den Zeilenendstil für Dateien mit gesetzter `svn:eol-style` Eigenschaft zu überschreiben.

Selbstverständlich können sie auch direkt aus einem Projektarchiv exportieren. Benutzen sie dazu den Projektarchivbetrachter, wählen den entsprechenden Pfad und benutzen dann das Kontextmenü → Exportieren. Ihnen wird dann der oben beschriebene Exportiere von URL-Dialog angezeigt.

Wenn Sie den Befehl in einer Arbeitskopie ausführen werden Sie nach einem Ort gefragt an dem Sie eine *saubere* Kopie Ihrer Arbeitskopie, ohne die `.svn` Ordner speichern möchten. Standardmäßig werden nur die versionierten Dateien exportiert, aber Sie können mit der Option Exportiere alle Dateien festlegen, dass auch die nicht versionierten Dateien exportiert werden sollen. Externe Referenzen via `svn:externals` können, wenn gewünscht, ausgelassen werden.

Eine andere Möglichkeit zu exportieren geht durch Rechts-Ziehen eines Arbeitskopieordners an einen anderen Ort. Dann einfach Kontextmenü → SVN hierher exportieren oder Kontextmenü → SVN alles hierher exportieren wählen. Die zweite Option exportiert auch die nicht versionierten Dateien.

Falls Sie eine Arbeitskopie exportieren und der Zielordner bereits einen Ordner mit demselben Namen enthält, wie derjenige, den sie gerade exportieren, können Sie auf Nachfrage den existierenden Ordner überschreiben oder einen neuen Ordner mit einem (n) Suffix angelegen lassen, z.B. Ziel (1).



Einzelne Dateien exportieren

Aus dem Exportdialog heraus können keine einzelnen Dateien exportiert werden, obwohl Subversion dazu in der Lage ist.

Um einzelne Dateien mit TortoiseSVN zu exportieren, müssen Sie den Projektarchivbetrachter verwenden ([Abschnitt 4.24, „Projektarchivbetrachter“](#)). Ziehen Sie einfach die Datei(en) mit Hilfe der Maus aus dem Projektarchivbetrachter in den Windows Explorer oder exportieren Sie die Datei über das Kontextmenü.



Exportieren von Baumänderungen

Falls Sie eine Kopie Ihrer Projektstruktur exportieren wollen, die nur die zwischen zwei Revisionen geänderten Objekte enthält, verwenden Sie die in [Abschnitt 4.10.3, „Ordner vergleichen“](#) beschriebene Funktion zum Vergleichen von Revisionen.

4.26.1. Eine Arbeitskopie aus der Versionskontrolle entfernen

Manchmal möchten Sie eine Arbeitskopie in einen nicht versionierten Ordner ohne die .svn Verzeichnisse zurückverwandeln. Was Sie bräuchten wäre ein Befehl, der die Steuerverzeichnisse entfernt ohne einen zusätzlichen, sauberen Dateibaum zu erzeugen.

Die Antwort ist überraschend einfach - Exportieren Sie den Ordner auf in selbst! TortoiseSVN kennt diesen Sonderfall und fragt nach, ob Sie die Arbeitskopie aus der Versionskontrolle entfernen wollen. Wenn Sie mit *Ja* bestätigen, werden die Steuerverzeichnisse entfernt und sie werden einen leeren, nicht versionierten Dateibaum erhalten.

4.27. Eine Arbeitskopie umplatzieren

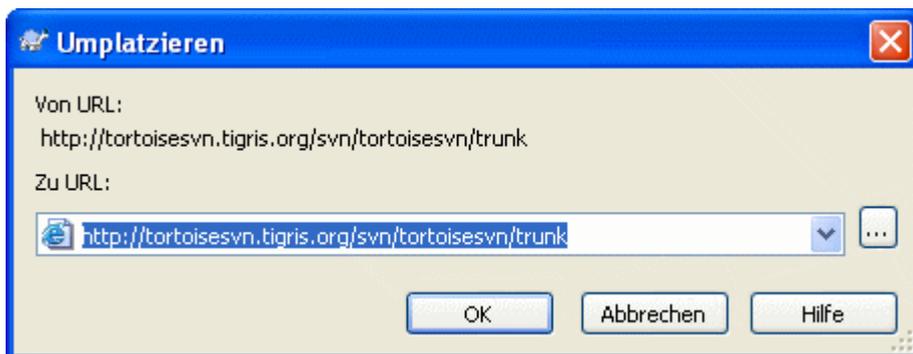


Abbildung 4.49. Der Umplatzieren-Dialog

Sollte die URL Ihres Projektarchivs sich geändert haben (z.B. der DNS-Eintrag des Servers hat sich geändert), so müssen Sie nicht erneut eine ganze Arbeitskopie aus dem Projektarchiv auschecken. Es reicht wenn Sie den Befehl TortoiseSVN → Umplatzieren... ausführen und die neue URL angeben. Dieser Befehl durchläuft alle `entries` Dateien im `.svn` Verzeichnis und trägt dort die neue URL ein.

Es mag Sie überraschen, dass TortoiseSVN als Teil dieser Operation das Projektarchiv kontaktiert. Es müssen ein paar einfache Tests durchgeführt werden, um sicherzustellen, dass die neue URL wirklich auf das zur Arbeitskopie gehörige Projektarchiv verweist.



Warnung

Dies ist eine sehr selten durchgeführte Aktion. Der Umplatzieren Befehl darf nur angewendet werden, wenn sich die URL der Arbeitskopie geändert hat. Mögliche Gründe dafür sind:

- Die IP Adresse des Servers hat sich geändert.

- Das Protokoll hat sich geändert (z.B. von http:// in https://).
- Der Basispfad des Projektarchivs hat sich in der Serverkonfiguration geändert.

Andersherum ausgedrückt: Sie müssen Ihre Arbeitskopie umplatzieren, wenn diese weiterhin auf die gleiche Stelle im selben Projektarchiv zeigt wie vorher, das Projektarchiv sich aber nun woanders befindet.

Es darf nicht umplatziert werden, wenn:

- Sie zu einem anderen Subversion Projektarchiv wechseln wollen. In diesem Fall sollten Sie eine frische Arbeitskopie auschecken.
- Sie zu einem anderen Zweig oder Verzeichnis innerhalb desselben Projektarchivs wechseln wollen. Dazu müssen Sie den TortoiseSVN → Wechseln zu... Befehl verwenden, der in [Abschnitt 4.19.2, „Auschecken oder Wechseln...“](#) beschrieben wird.

Wenn Sie fälschlicherweise in einem der oben genannten Fälle Ihre Arbeitskopie umplatzieren, *wird diese ruiniert!* Sie werden unerklärliche Fehlermeldungen beim Aktualisieren, Übertragen usw. erhalten. Sobald diese geschehen ist, müssen Sie Ihre Arbeitskopie frisch auschecken.

4.28. Integration mit einem System zur Fehlerverfolgung

Es ist in der Software Entwicklung üblich, Änderungen am Quellcode mit einer spezifischen ID in einem Fehlerverfolgungssystem zu verbinden. Dies wird meist mittels eines `pre-commit` Aktionsskripts im Projektarchiv erreicht, das vom Fehlerverfolgungssystem zur Verfügung gestellt wird und die Logmeldung einer Übertragung nach bestimmten Wörtern durchsucht. Dies ist jedoch sehr fehleranfällig, da der Benutzer sich merken muss wie die Logmeldung auszusehen hat damit das Aktionsskript dies auch richtig erkennen kann.

TortoiseSVN kann hier zweifach helfen:

1. Wenn der Benutzer eine Logmeldung eingibt, kann eine vordefinierte Zeile automatisch der Logmeldung angehängt oder vorangestellt werden. Dies reduziert das Risiko, dass der Benutzer die Logmeldung falsch eingibt und das Aktionsskript die Eintragsnummer nicht mehr erkennen kann.

Oder TortoiseSVN kann, während der Benutzer die Logmeldung eingibt die Stelle in der Logmeldung hervorheben, die vom Aktionsskript als Eintragsnummer erkannt wird. Auf diese Weise hat der Benutzer eine visuelle Bestätigung für die Korrektheit der Logmeldung.

2. Wenn der Benutzer die Logmeldungen im Log-Dialog ansieht, kann TortoiseSVN einen Verweis zum Eintrag erstellen und damit durch einen Klick den Web-Browser gleich auf die entsprechende Seite führen.

4.28.1. Eintragsnummern in Logmeldungen einfügen

Sie können ein Fehlerverfolgungssystem Ihrer Wahl in TortoiseSVN einbinden. Dafür müssen Sie ein paar Subversion Eigenschaften definieren, die mit `bugtraq:` beginnen. Diese Eigenschaften müssen auf Ordner gesetzt werden: ([Abschnitt 4.17, „Projekt-Einstellungen“](#))

Es gibt zwei verschiedene Wege, um TortoiseSVN mit einem Fehlerverfolgungssystem zu verbinden. Ein Weg basiert auf einfachen Zeichenketten, der andere basiert auf *Regulären Ausdrücken*. Die Eigenschaften, welche von beiden Methoden benutzt werden sind:

`bugtraq:url`

Hier geben Sie die URL zu Ihrem Fehlerverfolgungssystem ein. Die URL muss korrekt URI-codiert sein und `%BUGID%` enthalten. `%BUGID%` wird später durch die Eintragsnummer ersetzt. Dies ermöglicht es TortoiseSVN einen direkten Verweis auf die Revision im Log-Dialog anzuzeigen,

so dass Sie direkt dorthin springen können. Sie müssen diese Eigenschaft nicht setzen, aber wenn Sie sie weglassen, zeigt TortoiseSVN später nur die Eintragsnummer an und keinen Verweis. Ein Beispiel für TortoiseSVN wäre `http://issues.tortoisesvn.net/?do=details&id=%BUGID%`

Sie können auch relative URLs anstelle von absoluten verwenden. Das ist nützlich, falls Ihr Fehlerverfolgungssystem sich in derselben Domäne oder auf demselben Server befindet, wie Ihr Projektarchiv. Falls sich der Domänenname ändern sollte, müssen Sie die `bugtraq:url` Eigenschaft nicht anpassen. Es gibt zwei Möglichkeiten, relative URLs zu definieren:

Falls die URL mit der Zeichenkette `^/` beginnt, wird sie relativ zur Wurzel des Projektarchivs angenommen. Zum Beispiel wird `^/../?do=details&id=%BUGID%` zu `http://tortoisesvn.net/?do=details&id=%BUGID%` aufgelöst, falls Ihr Projektarchiv sich auf `http://tortoisesvn.net/svn/trunk/` befindet.

Eine URL die mit der Zeichenkette `/` anfängt, wird relative zum Namen des Servers interpretiert. Zum Beispiel wird `/?do=details&id=%BUGID%` zu `http://tortoisesvn.net/?do=details&id=%BUGID%` aufgelöst, falls Ihr Projektarchiv sich irgendwo auf `http://tortoisesvn.net` befindet.

`bugtraq:warnifnoissue`

Wenn die Eigenschaft auf `true` gesetzt ist, warnt Sie TortoiseSVN, falls das Feld für die Eintragsnummer im Übertragen-Dialog leer gelassen wurde. Das ist nur eine Erinnerungshilfe, falls der Benutzer die Eingabe vergessen hat. Gültige Werte: `true/false`. Wenn keine Eigenschaft definiert ist, wird `false` angenommen.

4.28.1.1. Eintragsnummer in einem Textfeld

In der einfachen Variante zeigt TortoiseSVN dem Benutzer ein zusätzliches Eingabefeld für die Eintragsnummer an. Es wird dann der Logmeldung eine zusätzliche Zeile angehängt oder vorangestellt.

`bugtraq:message`

Diese Eigenschaft aktiviert das Fehlerverfolgungssystem im *Eingabefeld* Modus. Wenn sie gesetzt ist, wird TortoiseSVN Sie auffordern, beim Übertragen eine Eintragsnummer anzugeben. Sie wird benutzt, um eine extra Zeile an das Ende der Logmeldung anzufügen. Dafür muss sie den Text `%BUGID%` enthalten, der beim Übertragen durch die Eintragsnummer ersetzt wird. Mithilfe dieses Textes kann Ihr Fehlerverfolgungssystem die Logmeldungen analysieren und die Revisionen einem Eintrag zuordnen. TortoiseSVN benutzt z.B. `Issue : %BUGID%`, aber das hängt von Ihrem Programm ab.

`bugtraq:append`

Diese Eigenschaft definiert ob die Eintragsnummer an das Ende angehängt oder vor der Logmeldung eingefügt wird. Gültige Werte sind: `true/false`. Wenn keine Eigenschaft definiert ist, wird `true` angenommen.

`bugtraq:label`

Dieser Text wird von TortoiseSVN im Übertragen-Dialog angezeigt. Er beschreibt das Eingabefeld für die Eintragsnummer. Wird er weggelassen, wird `Fehler-ID / Eintrags-Nr:` angezeigt. Beachten Sie bitte, dass der Text nicht zu lang ist, da er sonst eventuell einfach abgeschnitten wird. (empfohlen werden max 20 bis 25 Zeichen)

`bugtraq:number`

Auf `true` gesetzt, werden, außer dem Komma nur Zahlen im Fehlerverfolgungs Eingabefeld akzeptiert. `False` erlaubt auch die Eingabe von Buchstaben. Gültige Werte: `true/false`. Wenn keine Eigenschaft definiert ist, wird `true` angenommen.

4.28.1.2. Eintragsnummern mittels regulärer Ausdrücke

In der Variante mit den *Regulären Ausdrücken* zeigt TortoiseSVN kein separates Eingabefeld an, sondern hebt den Teil der Logmeldung hervor, der als Eintragsnummer erkannt wird. Die Erkennung erfolgt während der Benutzer die Logmeldung eingibt. Das bedeutet auch, dass die Eintragsnummer irgendwo in

der Logmeldung stehen kann und nicht nur in einer separaten Zeile. Diese wesentlich flexiblere Methode wird vom TortoiseSVN Projekt verwendet.

bugtraq:logregex

Diese Eigenschaft aktiviert das Fehlerverfolgungssystem im *RegEx* Modus. Sie enthält entweder einen oder zwei reguläre Ausdrücke, die durch einen Zeilenumbruch getrennt werden.

Wenn zwei Ausdrücke definiert sind, ist der erste Ausdruck ein Vorfilter, mit dem Zeichenfolgen, die Eintrags-IDs enthalten gefunden werden. Der zweite Ausdruck extrahiert dann die Eintrags-IDs aus dem Ergebnis des ersten. Das ermöglicht es Ihnen, Eintrags-IDs auf natürliche Art in die Beschreibung einfließen zu lassen, z.B.: „Diese Änderung erledigt die Fehler #23, #24 und #35“.

Wenn Sie, wie im obigen Beispiel, Zeichenketten in einer Logmeldung finden möchten, können Sie dazu die folgenden beiden regulären Ausdrücke verwenden: `[Ff]ehler:?(\\s*(,|und)? \\s*#\\d+)+` sowie `(\\d+)` als zweiten Ausdruck.

Der erste Ausdruck schneidet „Fehler #23, #24 und #35“ aus der umgebenden Logmeldung. Der zweite Ausdruck extrahiert die reinen Dezimalzahlen aus der Ausgabe des ersten Ausdrucks. Er gibt also „23“, „24“ und „25“ als Eintrags-IDs zurück.

Eine kleine Erklärung des ersten regulären Ausdrucks: Er sucht nach einem Text der mit „Fehler“, alternativ klein geschrieben, beginnt. Danach kann ein Doppelpunkt folgen. Darauf folgen eine oder mehrere Gruppen, bestehend aus null oder mehr führenden Leerzeichen, einem optionalen Komma oder „und“ sowie möglicherweise weiteren Leerzeichen. Abschließend werden ein „#“ und eine Dezimalzahl gesucht.

Wenn nur ein regulärer Ausdruck gesetzt ist, dann müssen alleine mit ihm die Eintragsnummern in den Gruppen des Ausdrucks gefunden werden. Als Beispiel: `[Ii]ssue(?:s)? #?(\\d+)`. Diese Methode wird von einigen Fehlerverfolgungssystemen, wie z.B. Trac verwendet, es ist jedoch schwieriger, einen passenden regulären Ausdruck zu konstruieren. Wir empfehlen, dass Sie diese Methode nur verwenden, wenn Ihr Fehlerverfolgungssystem dies erfordert.

Wenn Sie sich mit regulären Ausdrücken noch nicht so gut auskennen, lesen Sie sich die Einführung unter http://de.wikipedia.org/wiki/Regulärer_Ausdruck [http://de.wikipedia.org/wiki/Regul%C3%A4rer_Ausdruck/] sowie die Online-Dokumentation und -Anleitung auf <http://www.regular-expressions.info/> [<http://www.reular-expressions.info/>] durch.

Wenn sowohl die `bugtraq:message` als auch die `bugtraq:logregex` Eigenschaft gesetzt ist, erhält `logregex` Vorrang.



Tipp

Auch wenn Sie gar kein Fehlerverfolgungssystem mit einem pre-commit Aktionskript haben, können Sie diese Funktion dazu nutzen, Eintragsnummern welche Sie in Ihren Logmeldungen erwähnt haben, in einen Verweis zum Fehlerverfolgungssystem zu verwandeln.

Auch wenn Sie die Verweise selbst nicht benötigen, können Sie Änderungen zu einer bestimmten Nummer einfach finden, denn die Eintragsnummern werden in separaten Spalten im Log-Dialog angezeigt.

Manche `tsvn:` Eigenschaften erfordern einen `true/false` Wert. TortoiseSVN versteht auch `yes` als Synonym für `true` und `no` als Synonym für `false`.



Setzen Sie die Eigenschaften auf Ordner

Die obigen Eigenschaften müssen auf Ordner gesetzt sein, damit das System richtig funktioniert. Beim Übertragen werden die Eigenschaften des aktuellen Ordners gelesen.

Falls die Eigenschaften nicht gefunden werden, sucht TortoiseSVN nach oben durch die Ordnerstruktur bis es auf einen nicht versionierten Ordner oder die Wurzel (z.B. C:\) stößt. Wenn Sie sicher gehen können, dass jeder Benutzer z.B. von trunk/ und nicht aus einem Unterordner auscheckt, definieren Sie die Eigenschaften nur für trunk/. Wenn Sie nicht sicher sein können, definieren Sie die Eigenschaft rekursiv für jeden Unterordner. Eigenschaften in einem Unterordner überschreiben die Einstellungen in einem übergeordneten Ordner.

Ausschließlich für tsvn: Eigenschaften können Sie die Rekursiv Option wählen, um die Eigenschaft auf alle Unterordner zu setzen, ohne sie gleichzeitig auch für alle Dateien zu setzen.



Keine Fehlerverfolgungsinformationen im Projektarchivbetrachter

Da die Integration mit Fehlerverfolgungssystemen vom Zugriff auf Subversion Eigenschaften abhängt, sehen Sie die Ergebnisse nur in einer Arbeitskopie. Im Projektarchivbetrachter steht diese Funktion nicht zur Verfügung, weil das Holen der Eigenschaften von einem Server eine zeitintensive Operation list.

Die Integration mit Fehlerverfolgungssystemen ist nicht auf TortoiseSVN limitiert, sie kann von jedem Subversion Client benutzt werden. Für weitere Informationen hierzu lesen Sie bitte die [Issuetracker Integration Specification](https://tortoisesvn.tigris.org/svn/tortoisesvn/trunk/doc/issuetrackers.txt) [tsvn:https://tortoisesvn.tigris.org/svn/tortoisesvn/trunk/doc/issuetrackers.txt] im TortoiseSVN Projektarchiv. ([Abschnitt 3](#), „TortoiseSVN ist frei!“ erklärt, wie man auf das TortoiseSVN Projektarchiv zugreift).

4.28.2. Informationen vom Fehlerverfolgungssystem beziehen

Der vorherige Abschnitt befasste sich damit, Eintragsnummern zu den Logmeldungen hinzuzufügen. Aber wie bekommt man Informationen aus dem Fehlerverfolgungssystem? Der Übertragen-Dialog besitzt eine Windows COM Schnittstelle mit deren Hilfe ein externes Programm eingebunden werden kann, das mit dem Fehlerverfolgungssystem kommuniziert. Ein typischer Fall wäre, das Fehlerverfolgungssystem nach einer Liste der Ihnen zugeordneten offenen Aufgaben zu fragen, die Sie beim Übertragen als erledigt markieren wollen.

Jede solche Schnittstelle ist natürlich sehr spezifisch für Ihr System, so dass wir diesen Teil nicht zur Verfügung stellen können und die Beschreibung eines solchen Programms sprengt den Rahmen dieses Handbuchs. Die Schnittstellendefinition und Beispielanwendungen können Sie im contrib Verzeichnis des [TortoiseSVN Projektarchivs](http://tortoisesvn.googlecode.com/svn/trunk/contrib/issue-tracker-plugins) [http://tortoisesvn.googlecode.com/svn/trunk/contrib/issue-tracker-plugins] finden. ([Abschnitt 3](#), „TortoiseSVN ist frei!“ erklärt, wie man auf das TortoiseSVN Projektarchiv zugreift). Eine Zusammenfassung der API findet sich in [Kapitel 6, IBugtraqProvider Schnittstelle](#). Ein weiteres, funktionierendes Beispiel in C# ist [Gurtle](http://code.google.com/p/gurtle/) [http://code.google.com/p/gurtle/], das die COM Schnittstelle für den Zugriff auf das [Google Code](http://code.google.com/hosting/) [http://code.google.com/hosting/] Fehlerverfolgungssystem implementiert.

Lassen Sie uns zur Veranschaulichung annehmen, dass Ihr Systemadministrator Sie mit einem Modul für Ihr Fehlerverfolgungssystem versorgt hat, welches Sie installiert haben. Ferner haben Sie im TortoiseSVN Einstellungsdialog einige Ihrer Arbeitskopien so eingerichtet, dass diese die Integration der Fehlerverfolgung nutzen. Wenn Sie nun den Übertragen-Dialog in einer der Arbeitskopien aufrufen, denen das Modul zugewiesen ist, sehen Sie oben im Dialog eine neue Schaltfläche.

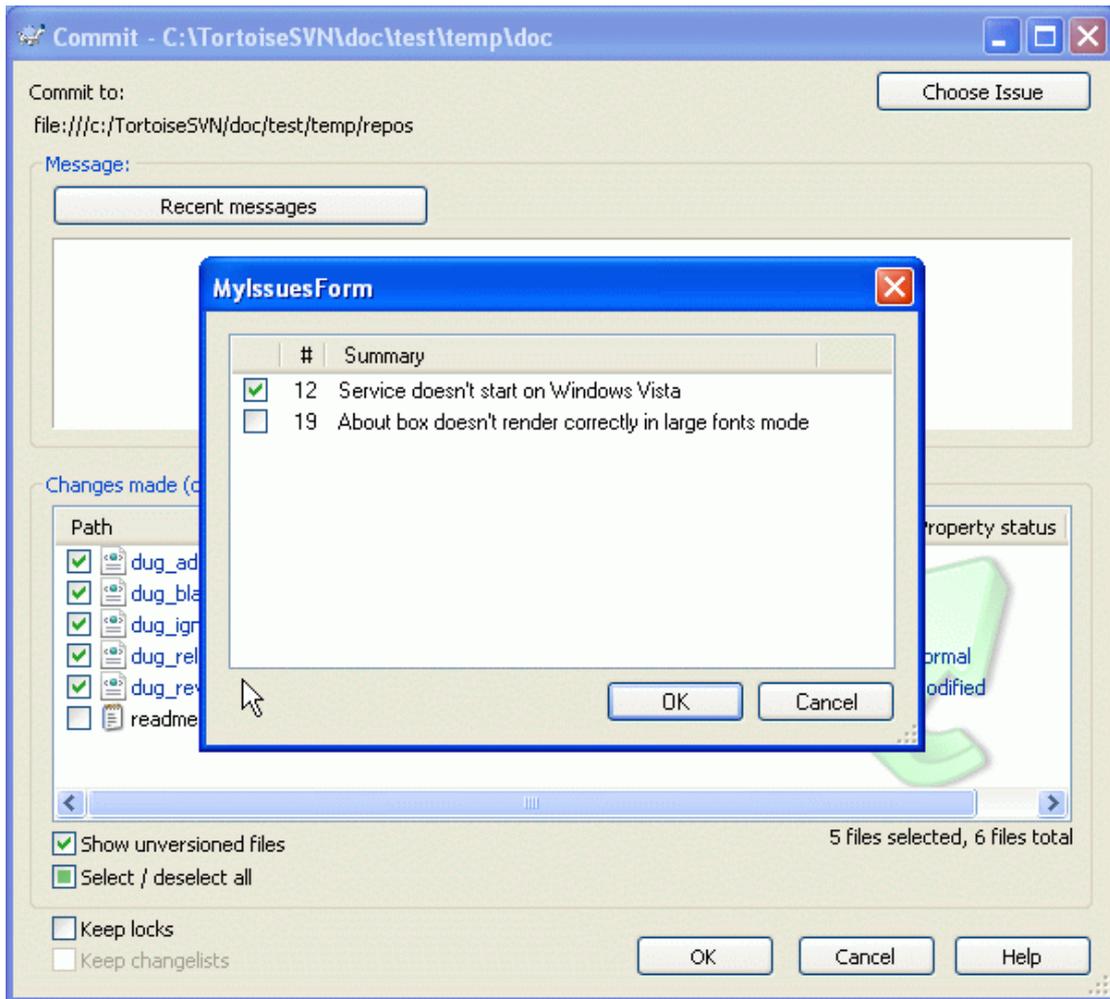


Abbildung 4.50. Beispielabfrage des Fehlerverfolgungssystems

In diesem Beispiel können Sie eine oder mehrere offene Aufgaben auswählen. Das Modul kann dann automatisch Text für Ihre Logmeldung generieren.

4.29. Integration mit webbasierten Projektarchivbetrachtern

Es gibt mehrere webbasierte Projektarchivbetrachter, die mit Subversion eingesetzt werden können, zum Beispiel: *ViewVC* [<http://www.viewvc.org/>] und *WebSVN* [<http://websvn.tigris.org/>]. TortoiseSVN bietet Ihnen eine Integrationsmöglichkeit mit diesen Betrachtern an.

Sie können einen Projektarchivbetrachter Ihrer Wahl in TortoiseSVN einbinden. Dafür müssen Sie ein paar Eigenschaften angeben, die den Verweis definieren. Diese Eigenschaften müssen für Ordner gesetzt werden: ([Abschnitt 4.17, „Projekt-Einstellungen“](#))

`webviewer:revision`

Setzen Sie diese Eigenschaft auf die URL Ihres Projektarchivbetrachters mit der Sie alle Änderungen einer bestimmten Revision sehen können. Sie muss korrekt URI codiert sein und `%REVISION%` enthalten. `%REVISION%` wird durch die gewünschte Revisionsnummer ersetzt. Dies ermöglicht TortoiseSVN im Log-Dialog ein Kontextmenü `Kontextmenü → Betrachte Revision im Web-Browser` anzuzeigen.

`webviewer:pathrevision`

Setzen Sie diese Eigenschaft auf die URL mit der Sie Änderungen an einer Datei in einer bestimmten Revision sehen können. Sie muss korrekt URI codiert sein sowie `%REVISION%` und `%PATH%`

enthalten. %PATH% wird durch den Pfad relativ zur Projektarchivbasis ersetzt. Dies ermöglicht TortoiseSVN im Log-Dialog ein Kontextmenü **Kontextmenü** → **Betrachte Revision eines Pfades im Web-Browser** anzuzeigen. Wenn Sie zum Beispiel im Log-Dialog auf eine Datei /trunk/src/file klicken, wird %PATH% in der URL durch /trunk/src/file ersetzt.

Sie können auch relative URLs anstelle von absoluten verwenden. Das ist nützlich, falls Ihr Web-Betrachter sich in derselben Domäne oder auf demselben Server befindet, wie Ihr Projektarchiv. Falls sich der Domänenname ändern sollte, müssen Sie die `webviewer:revision` und `webviewer:pathrevision` Eigenschaften nicht anpassen. Das Format ist dasselbe wie bei der `bugtraq:url` Eigenschaft und in [Abschnitt 4.28](#), „Integration mit einem System zur Fehlerverfolgung“ erklärt.



Setzen Sie die Eigenschaften auf Ordner

Die obigen Eigenschaften müssen auf Ordner gesetzt sein, damit das System richtig funktioniert. Beim Übertragen werden die Eigenschaften des aktuellen Ordners gelesen. Falls die Eigenschaften nicht gefunden werden, sucht TortoiseSVN nach oben durch die Ordnerstruktur bis es auf einen nicht versionierten Ordner oder die Wurzel (z.B. C:\) stößt. Wenn Sie sicher gehen können, dass jeder Benutzer z.B. von trunk/ und nicht aus einem Unterordner auscheckt, definieren Sie die Eigenschaften nur für trunk/. Wenn Sie nicht sicher sein können, definieren Sie die Eigenschaft rekursiv für jeden Unterordner. Eigenschaften in einem Unterordner überschreiben die Einstellungen in einem übergeordneten Ordner.

Ausschließlich für `tsvn`: Eigenschaften können Sie die **Rekursiv** Option wählen, um die Eigenschaft auf alle Unterordner zu setzen, ohne sie gleichzeitig auch für alle Dateien zu setzen.



Keine Projektarchiv-Verweise im Projektarchivbetrachter

Da die Integration des Projektarchivbetrachters vom Zugriff auf Subversion Eigenschaften abhängt, sehen Sie die Ergebnisse nur in einer Arbeitskopie. Im Projektarchivbetrachter steht diese Funktion nicht zur Verfügung, weil das Holen der Eigenschaften von einem Server eine zeitintensive Operation ist.

4.30. TortoiseSVN Einstellungen

Um zusätzliche Hilfe für bestimmte Einstellungen zu erhalten, lassen Sie den Mauszeiger eine Sekunde lang über den Eingabefeldern, Optionsfeldern, ... und es wird ein kleiner Tipp erscheinen.

4.30.1. Allgemeine Einstellungen

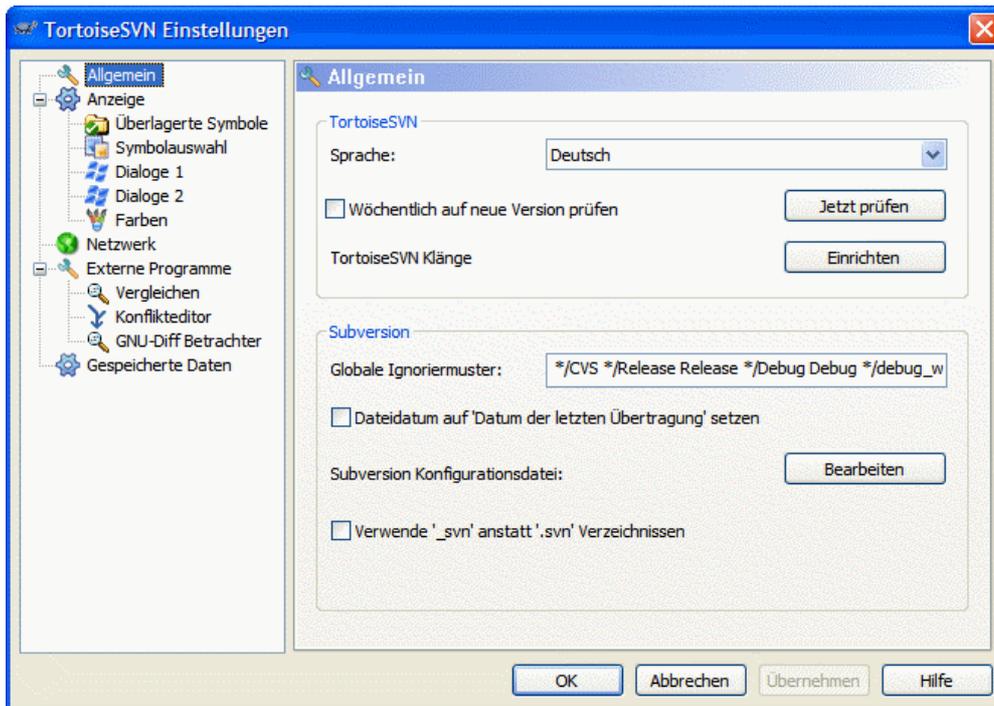


Abbildung 4.51. Der Einstellungsdialog, Allgemein

Dieser Dialog erlaubt Ihnen die Sprache der Anwenderoberfläche sowie einige Subversion Parameter einzustellen.

Sprache

Wählt die Sprache für die Dialoge/Meldungen aus. Was haben Sie anderes erwartet?

Prüfe jede Woche auf neue Versionen

Wenn diese Option aktiviert ist, wird TortoiseSVN einmal pro Woche seine Downloadseite überprüfen, ob eine neue Version zur Verfügung steht. Mittels **Jetzt prüfen** können Sie eine sofortige Prüfung veranlassen. Die neue Version wird nicht herunter geladen. Sie erhalten lediglich Auskunft darüber, ob Ihre derzeitige Version noch aktuell ist.

TortoiseSVN Klänge

TortoiseSVN liefert drei eigene Klänge mit, die automatisch installiert werden.

- Fehler
- Hinweis
- Warnung

Sie können andere Klänge in der Windows Systemsteuerung unter *Sounds und Audiogeräte* festlegen oder dort auch deaktivieren. Die Schaltfläche **Einrichten** bietet eine Abkürzung zur Systemsteuerung.

Globale Ignoriermuster

Globale Ignoriermuster sorgen dafür, dass bestimmte unversionierte Dateien nicht angezeigt werden, z.B. im Übertragen-Dialog. Außerdem werden solche Dateien beim Importieren in ein Projektarchiv ignoriert. Schließen Sie Dateien oder Ordner durch Angabe von Dateinamen oder Erweiterungen aus. Die einzelnen Muster werden durch Leerzeichen voneinander getrennt. Zum Beispiel `bin obj *.bak *.~?? *.jar *.[Tt]mp`. Diese Muster dürfen keine Pfadtrennzeichen enthalten.

Beachten Sie auch, dass es keine Möglichkeit gibt, zwischen Dateien und Ordnern zu unterscheiden. Lesen Sie weitere Informationen zur Syntax in [Abschnitt 4.13.1, „Platzhalter in der Ignorieren-Liste“](#) nach.

Beachten Sie, dass die Ignoriermuster, die Sie hier einstellen, auch andere Subversion Clients, inklusive der Subversion Kommandozeile, auf Ihrem PC beeinflussen.



Achtung

Wenn Sie die Subversion Konfigurationsdatei verwenden, um ein `global-ignores` Muster zu definieren, wird das die TortoiseSVN Einstellungen überlagern. Die Subversion Konfigurationsdatei kann über die **Bearbeiten** Schaltfläche geändert werden.

Dieses Ausschluss-Muster beeinflusst alle Ihre Projekte. Es ist nicht versioniert, also wird es keine anderen Anwender beeinflussen. Im Gegensatz dazu können Sie auch die versionierte `svn:ignore` Eigenschaft verwenden, um Dateien oder Verzeichnisse von der Versionskontrolle auszuschließen. Lesen Sie [Abschnitt 4.13, „Ignorieren von Dateien und Ordnern“](#) für weitere Information.

Dateidatum auf „Datum der letzten Übertragung“ setzen

Diese Einstellung zwingt TortoiseSVN das Dateidatum beim Auschecken oder Aktualisieren auf das Datum der letzten Übertragung der Datei zu setzen. Ansonsten verwendet TortoiseSVN das aktuelle Datum. Wenn Sie Software entwickeln, ist es normalerweise am besten, das aktuelle Datum zu verwenden, da Systeme zum automatischen Erstellen von Software normalerweise auf die Zeitstempel zurückgreifen, um zu entscheiden, ob eine Datei übersetzt werden soll. Wenn Sie das „Datum der letzten Übertragung“ verwenden und auf eine ältere Dateiversion zurückgreifen, kann es passieren, dass sich das Projekt nicht so übersetzen lässt, wie erwartet.

Subversion Konfigurationsdatei

Mittels **Bearbeiten** können Sie die Subversion-Konfigurationsdatei in einem Standard-Texteditor öffnen. Einige Einstellungen können nicht mit TortoiseSVN vorgenommen werden sondern nur direkt in der Konfigurationsdatei geändert werden. Für mehr Informationen über die Subversion `config` Datei lesen Sie bitte das Kapitel [Runtime Configuration Area](#) [<http://svnbook.red-bean.com/en/1.5/svn.advanced.confarea.html>] des Subversion Buchs. Der Abschnitt [Automatic Property Setting](#) [<http://svnbook.red-bean.com/en/1.5/svn.advanced.props.html#svn.advanced.props.auto>] ist hier auch von Interesse. Beachten Sie bitte, dass Subversion seine Einstellungen nacheinander an verschiedenen Stellen sucht. Die Reihenfolge und die Prioritäten werden in [Configuration and the Windows Registry](#) [<http://svnbook.red-bean.com/en/1.5/svn.advanced.confarea.html#svn.advanced.confarea.windows-registry>] erklärt.

Verwende `_svn` anstelle von `.svn` Verzeichnissen

VS.NET kann, wenn es mit Webprojekten genutzt wird, nicht mit den `.svn` Ordnern umgehen, die Subversion verwendet, um seine internen Informationen abzulegen. Das ist kein Fehler in Subversion. Der Fehler liegt in VS.Net und den von ihm verwendeten Frontpage Erweiterungen. Lesen Sie [Abschnitt 4.30.11, „Subversion Arbeitsordner“](#) für weitere Informationen zu diesem Thema.

Wenn Sie das Verhalten von Subversion und TortoiseSVN verändern wollen können Sie mit dieser Option die Umgebungsvariable setzen, die das Verhalten kontrolliert.

Beachten Sie dass das Ändern dieser Option nicht automatisch existierende Arbeitskopien auf den neuen Administrationsordner umstellt. Sie müssen das selbst mit Hilfe eines Skripts erledigen (Siehe FAQ) oder eine neue Arbeitskopie auschecken.

4.30.1.1. Einstellungen für das Kontextmenü

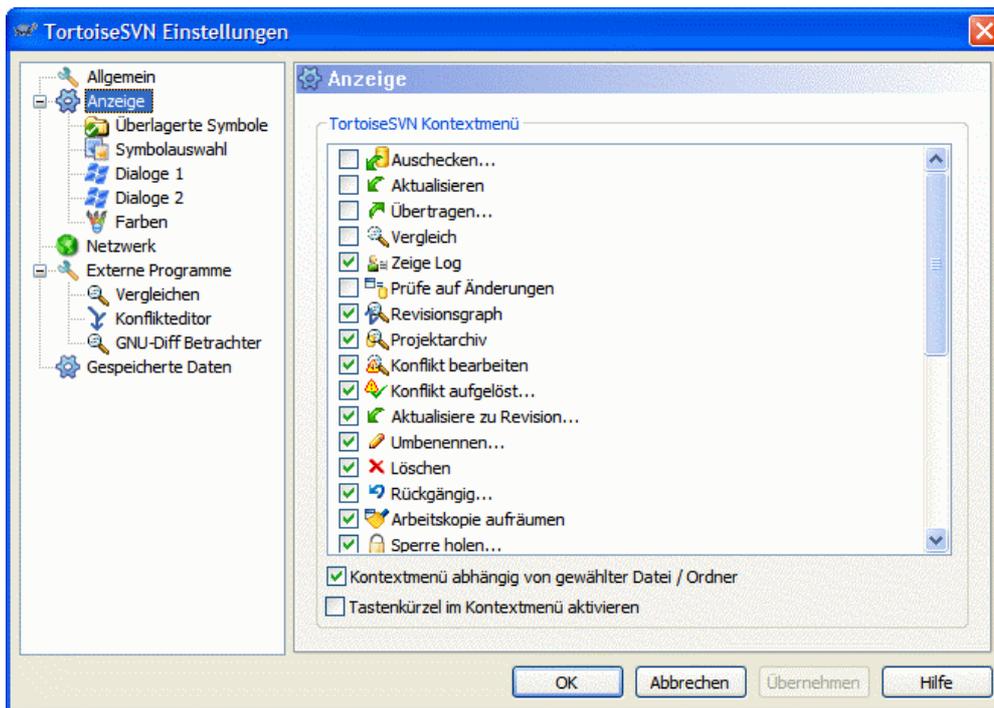


Abbildung 4.52. Der Einstellungsdialog, Kontextmenü

Auf dieser Seite können Sie einstellen, welche Einträge des TortoiseSVN Kontextmenüs im Hauptmenü und welche im TortoiseSVN Untermenü erscheinen sollen. Standardmäßig sind die meisten Einträge abgewählt und erscheinen im Untermenü.

Es gibt einen Sonderfall für **Hole Sperre**. Sie können diese Funktion natürlich in die oberste Ebene verschieben, aber da die meisten Dateien keine Sperre benötigen, wird es dadurch nur unübersichtlich. Andererseits erfordert eine Datei mit der `svn:needs-lock` diese Aktion jedes Mal, weshalb die Funktion in diesem Falle auf der obersten Ebene nützlich wäre. Wenn Sie diese Option markieren, wird, sobald eine Datei mit der `svn:needs-lock` Eigenschaft markiert wird, **Hole Sperre** in der obersten Menüebene angezeigt.

Wenn es Pfade auf Ihrem Computer gibt, in denen das TortoiseSVN Kontextmenü nicht angezeigt werden soll, geben Sie diese im unteren Eingabefeld an.

4.30.1.2. TortoiseSVN Dialoge Seite 1

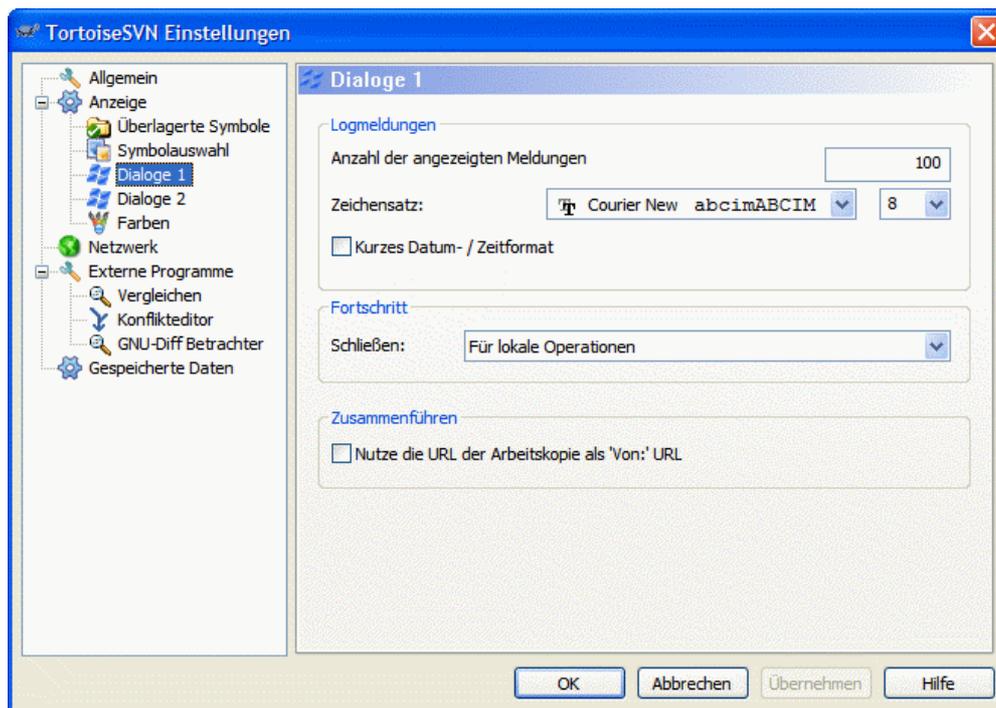


Abbildung 4.53. Einstellungen Dialoge, Seite 1

In diesem Dialog können Sie das Verhalten einiger TortoiseSVN Dialoge einstellen.

Anzahl der angezeigten Logmeldungen

Beschränkt die Anzahl der angezeigten Logmeldungen, welche TortoiseSVN vom Projektarchiv holt, wenn Sie TortoiseSVN → Zeige Log zum ersten Mal aufrufen. Nützlich bei langsamen Verbindungen zum Projektarchiv. Sie können jederzeit mehr Logmeldungen mittels Nächste 100 oder Zeige Alle anzeigen lassen.

Zeichensatz für Logmeldungen

Legt die Schriftart und Größe fest, in der Logmeldungen im Log-Dialog angezeigt und im Übertragen-Dialog eingegeben werden.

Kurzes Datum / Zeit Format in Logmeldungen

Wenn die standardmäßig angezeigten langen Datums-/Zeitangaben zu viel Platz einnehmen, können Sie hier TortoiseSVN anweisen, das kurze Format zu verwenden.

Kann in der Log-Liste doppelklicken um mit der vorherigen Revision zu vergleichen

Wenn Sie häufig im oberen Bereich des Log-Dialogs Revisionen vergleichen, können Sie diese Aktion aktivieren, damit der Vergleich per Doppelklick erfolgt. Sie ist standardmäßig deaktiviert, weil das Erstellen des Vergleichs manchmal eine langwierige Operation ist und viele die Wartezeit bei einem versehentlichen Doppelklick vermeiden wollen.

Fortschrittsanzeige

TortoiseSVN kann den Fortschrittsdialog automatisch schließen, wenn der Befehl ohne Fehler ausgeführt wurde. Diese Einstellung erlaubt Ihnen die Bedingungen zu setzen um den Dialog automatisch zu schließen. Die Standard-Einstellung (empfohlen) Manuell schließen erlaubt es Ihnen alle Meldungen nach jeder Aktion genau zu studieren. Wenn Sie möchten, können Sie den Dialog auch automatisch schließen lassen wenn keine kritischen Ereignisse aufgetreten sind.

Schließen, falls kein Löschen, Hinzufügen oder Zusammenführen lässt den Dialog nur offen, wenn keine Dateien zusammengeführt, hinzugefügt oder gelöscht wurde. Auch Konflikte und Fehler sorgen dafür, dass der Dialog geöffnet bleibt.

Schließen, falls kein Löschen, Hinzufügen oder Zusammenführen für Lokale Operationen lässt den Dialog offen wie bei Schließen, falls kein Löschen, Hinzufügen oder Zusammenführen, aber nur für lokale Operationen wie zum Beispiel Hinzufügen oder Rückgängig machen. Für Operationen welche das Projektarchiv mit einbeziehen bleibt der Dialog offen.

Schließen, falls keine Konflikte schließt den Dialog automatisch wenn keine Konflikte oder Fehlermeldungen aufgetreten sind.

Schließen, falls keine Fehler schließt den Dialog automatisch, auch wenn Konflikte aufgetreten sind. Der Dialog bleibt jedoch offen wenn Fehler aufgetreten sind, die verhindert haben, dass Subversion die Aktion ausführen konnte, zum Beispiel wenn der Server nicht erreichbar oder die Arbeitskopie veraltet ist.

Verwende Papierkorb bei Rückgängig

Wenn Sie lokale Änderungen zurücknehmen, werden diese verworfen. TortoiseSVN bietet Ihnen ein zusätzliches Sicherheitsnetz an, indem es die veränderten Dateien in den Papierkorb verschiebt, bevor der Originalzustand wieder hergestellt wird.

Nutze die URL der Arbeitskopie als „Von:“ URL

Im Zusammenführen-Dialog wird normalerweise die Von: URL zwischen zwei Aufrufen gespeichert und als Vorgabe eingestellt. Manchmal möchten Anwender jedoch von vielen verschiedenen Punkten in der Projekthierarchie aus Daten zusammenführen und finden es einfacher, mit der URL der aktuellen Arbeitskopie als Vorgabe zu beginnen. Diese kann dann verändert werden, um auf einen parallelen Pfad oder einen Zweig zu verweisen.

Auschecken Standardpfad

Sie können einen Standardpfad zum Auschecken festlegen. Wenn Sie all Ihre Arbeitskopien in einem Ordner haben, können Sie diesen Pfad als Vorbelegung hier einstellen, so dass Sie nur noch den Ordnernamen anhängen müssen.

Auschecken Standard-URL

Sie können außerdem eine Standard-URL zum Auschecken festlegen. Wenn Sie häufiger Unterprojekte eines großen Projektarchivs bearbeiten, kann es nützlich sein, die URL mit dem Basispfad vorzubelegen, so dass Sie nur noch den Projektnamen anhängen müssen.

4.30.1.3. TortoiseSVN Dialoge Seite 2

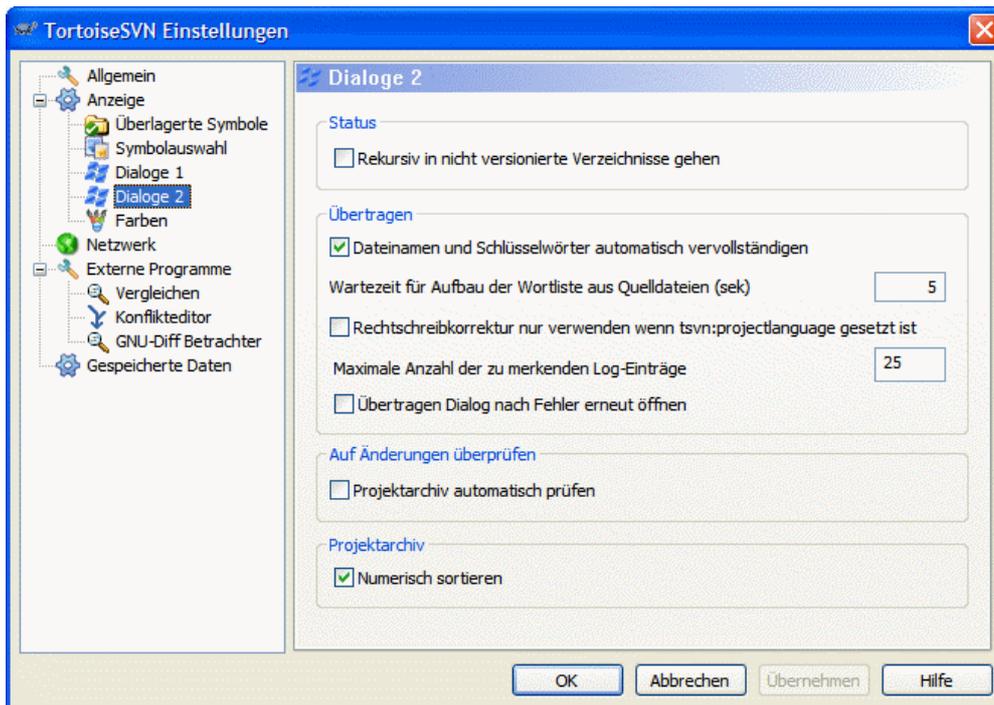


Abbildung 4.54. Einstellungen Dialoge, Seite 2

Rekursiv in nicht versionierte Verzeichnisse gehen

Wenn diese Option aktiviert ist (Vorgabe), werden jedes Mal, wenn ein nicht versionierter Ordner im „Hinzufügen“, „Übertragen“- oder Prüfe auf Änderungen-Dialog angezeigt wird, sämtliche Unterordner und Dateien dieses Ordners angezeigt. Sobald Sie die Option deaktivieren, wird nur der nicht versionierte Ordner angezeigt, was die Übersichtlichkeit in den Dialogen erhöht. Wenn Sie einen nicht versionierten Ordner zum Hinzufügen wählen, wird er in diesem Fall vollständig, rekursiv hinzugefügt.

Dateinamen und Schlüsselwörter automatisch Vervollständigen

Der Übertragen-Dialog bietet die Möglichkeit, die zu übertragenden Dateien nach Schlüsselwörtern zu durchsuchen. Wenn Sie die ersten drei Zeichen eines Wortes eingegeben haben, erscheint eine automatische Vervollständigungsliste, aus der Sie mittels **Enter** einen Eintrag auswählen können.

Wartezeit in Sekunden zum Stop der Datei-Analyse für das automatische Vervollständigen

Der Parser für das automatische Vervollständigen kann sehr langsam sein, wenn Sie sehr viele und große Dateien ausgewählt haben. Diese Wartezeit stoppt den Parser und übergibt die bisher gefundenen Daten an die Editor-Box. Wenn Sie wichtige Informationen für das automatische Vervollständigen vermissen, können Sie hier die Wartezeit erhöhen.

Rechtschreibkorrektur nur verwenden wenn `tsvn:projectlanguage` gesetzt ist

Wenn Sie die Rechtschreibkorrektur nicht benutzen möchten, aktivieren Sie diese Option. Die Rechtschreibkorrektur wird trotz dieser Einstellung aktiviert wenn das Projekt dies notwendig macht.

Maximale Anzahl zu merkender Log-Einträge

Wenn Sie eine Logmeldung im Übertragen Dialog eingeben, speichert TortoiseSVN standardmäßig die letzten 25 Logmeldungen für jedes Projektarchiv. Sie können die Anzahl hier festlegen. Wenn Sie viele verschiedene Projektarchive nutzen, sollten Sie die Zahl vielleicht heruntersetzen, um weniger Einträge in der Registrierung zu belegen.

Beachten Sie bitte, dass diese Einstellung nur Logmeldungen betrifft, die Sie auf diesem Computer eingeben. Sie hat nichts mit dem Log-Puffer zu tun.

Nach einer fehlgeschlagenen Übertragung die Übertragen- bzw. Verzweigen/Markieren-Dialoge erneut öffnen

Wenn eine Übertragung fehlschlägt (z.B. weil vorher die Arbeitskopie aktualisiert werden muss, ein Aktionsskript die Übertragung verwirft, ein Netzwerkfehler auftritt, etc.), können Sie mit dieser Option den Übertragen-Dialog für einen erneuten Versuch geöffnet halten. Sie sollten sich dessen bewusst sein, dass das auch zu Problemen führen kann. Falls zum Beispiel Ihre Arbeitskopie aktualisiert werden muss und die Aktualisierung zu Konflikten führt, müssen diese zunächst von Ihnen aufgelöst werden.

Objekte automatisch auswählen

Das normale Verhalten des Übertragen-Dialogs ist es, dass alle veränderten, versionierten Objekte zur Übertragung gewählt sind. Wenn Sie möchten, dass keine Vorauswahl getroffen wird, deaktivieren Sie diese Option.

Projektarchiv beim Start kontaktieren

Der Prüfe auf Änderungen-Dialog prüft in der Standardeinstellung die Arbeitskopie und kontaktiert das Projektarchiv nur, wenn Sie die Projektarchiv prüfen Schaltfläche betätigen. Wenn Sie das Projektarchiv immer überprüfen wollen, aktivieren Sie diese Option, damit das beim Start des Dialoges automatisch geschieht.

Zeige den Sperrdialog bevor Dateien gesperrt werden

Wenn Sie eine oder mehrere Dateien markieren und TortoiseSVN → Sperre holen... aufrufen, können Sie, wie in manchen Projekten üblich, im folgenden Dialog einen Grund für die Sperrung angeben. Sollten Sie nicht mit Sperrmeldungen arbeiten, können Sie diese Option abwählen, um den Dialog zu überspringen und die Dateien sofort zu sperren.

Wenn Sie TortoiseSVN → Sperre holen... auf einem Ordner aufrufen, wird der Dialog stets angezeigt, damit Sie die zu sperrenden Dateien wählen können.

Wenn Ihr Projekt die `tsvn:lockmsgminsize` Eigenschaft verwendet, wird der Sperrdialog stets, unabhängig von dieser Einstellung angezeigt, weil das Projekt Sperrmeldungen *erfordert*.

4.30.1.4. TortoiseSVN Farben

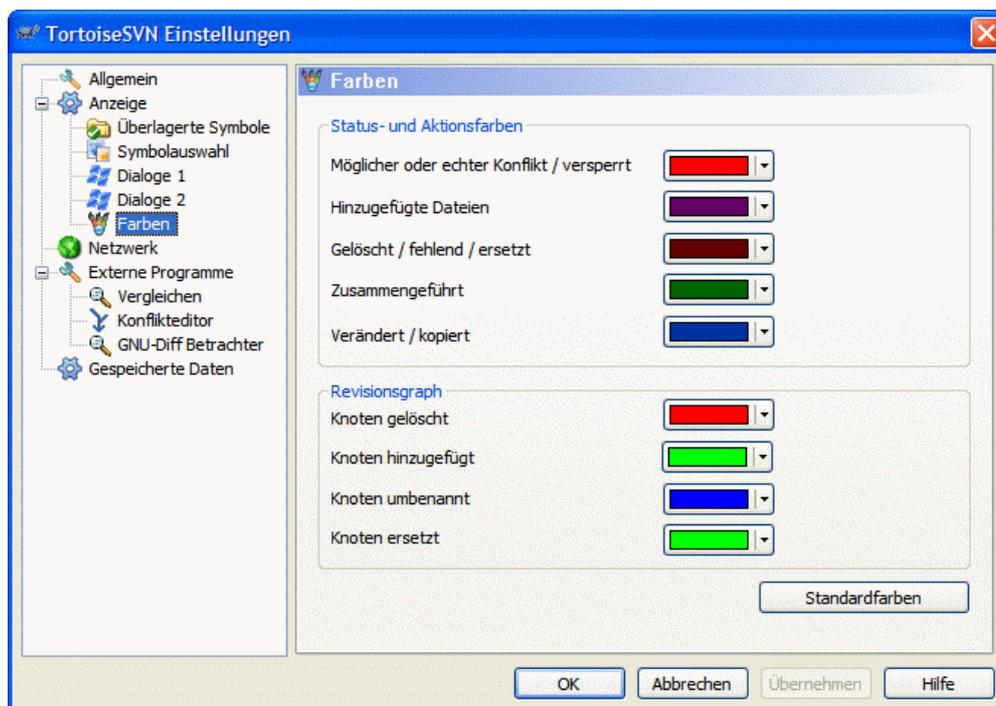


Abbildung 4.55. Der Einstellungsdialog, Farben

In diesem Dialog können Sie die Textfarben einiger TortoiseSVN Dialoge einstellen.

Möglicher oder echter Konflikt / versperrt

Ein Konflikt ist beim Aktualisieren aufgetreten oder kann beim Zusammenführen auftreten. Ein(e) nicht versionierte(r) Datei / Ordner versperrt die Aktualisierung eines gleichnamigen versionierten Objekts.

Diese Farbe wird auch für Fehlermeldungen im Fortschrittsdialog verwendet.

Hinzugefügte Dateien

Zum Projektarchiv hinzugefügte Objekte

Gelöscht / fehlend / ersetzt

Objekte, die aus dem Projektarchiv gelöscht wurden oder in der Arbeitskopie fehlen oder aus der Arbeitskopie gelöscht und durch ein anderes Objekt des gleichen Namens ersetzt wurden.

Zusammengeführt

Änderungen aus dem Projektarchiv die erfolgreich mit Ihren lokalen Änderungen zusammengeführt wurden, ohne einen Konflikt zu verursachen.

Verändert / kopiert

Mit Historie hinzugefügte bzw. im Projektarchiv kopierte Objekte. Wird auch im Log-Dialog für Einträge mit kopierten Objekten verwendet.

Knoten gelöscht

Ein Objekt, das aus dem Projektarchiv gelöscht wurde.

Knoten hinzugefügt

Ein Objekt, das durch Hinzufügen, Kopieren oder Verschieben zum Projektarchiv hinzugefügt wurde.

Knoten umbenannt

Ein Objekt, das im Projektarchiv umbenannt wurde.

Knoten ersetzt

Das originale Objekt wurde gelöscht und eines mit dem selben Namen ersetzt es.

4.30.2. Einstellungen des Revisionsgraphen

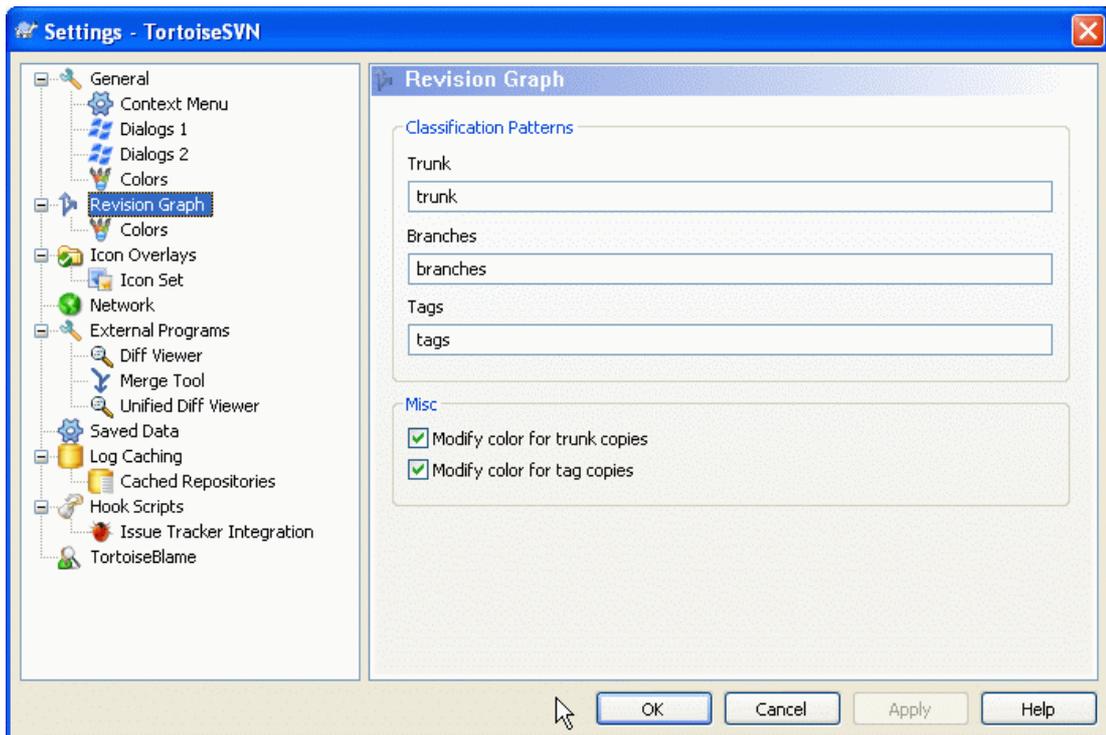


Abbildung 4.56. Der Einstellungsdialog, Revisionsgraph

Klassifizierungsmuster

Der Revisionsgraph versucht ein klareres Bild des Projektarchivs zu schaffen, indem er zwischen Stamm, Verzweigung und Marken unterscheidet. Da eine solche Unterscheidung nicht in Subversion eingebaut ist, wird die Information aus den Pfadnamen extrahiert. Die Standardeinstellung nimmt an, dass Sie die üblichen, in der Subversion angegebenen, englischen Namen verwenden. Sie können das selbstverständlich an Ihre Gepflogenheiten anpassen.

Geben Sie die Muster zur Klassifizierung der Pfade in den drei Eingabefeldern an. Die Muster werden ohne Berücksichtigung der Groß-/Kleinschreibung verglichen, sie müssen jedoch in Kleinbuchstaben angegeben werden. Die Platzhalter * und ? funktionieren wie gewohnt und Sie können zusätzlich ; zum Trennen mehrerer Muster angeben. Verwenden Sie keine Leerzeichen, da diese Bestandteil des Mustervergleichs werden.

Farben ändern

Farben werden im Revisionsgraphen verwendet, um den Knotentypen (hinzugefügt, gelöscht, umbenannt) zu kennzeichnen. Damit Sie gleichzeitig Klassifizierungen und Knotentypen erkennen können, kann der Revisionsgraph die beiden Farben überlagern. Wenn Sie diese Optionen aktivieren, wird die Überlagerung der Farben verwendet. Wenn die Option nicht aktiv ist, wird nur die Farbe für den Knotentypen verwendet. Im Farbeinstellungsdialog können Sie die gewünschten Farben festlegen.

4.30.2.1. Farben des Revisionsgraphen

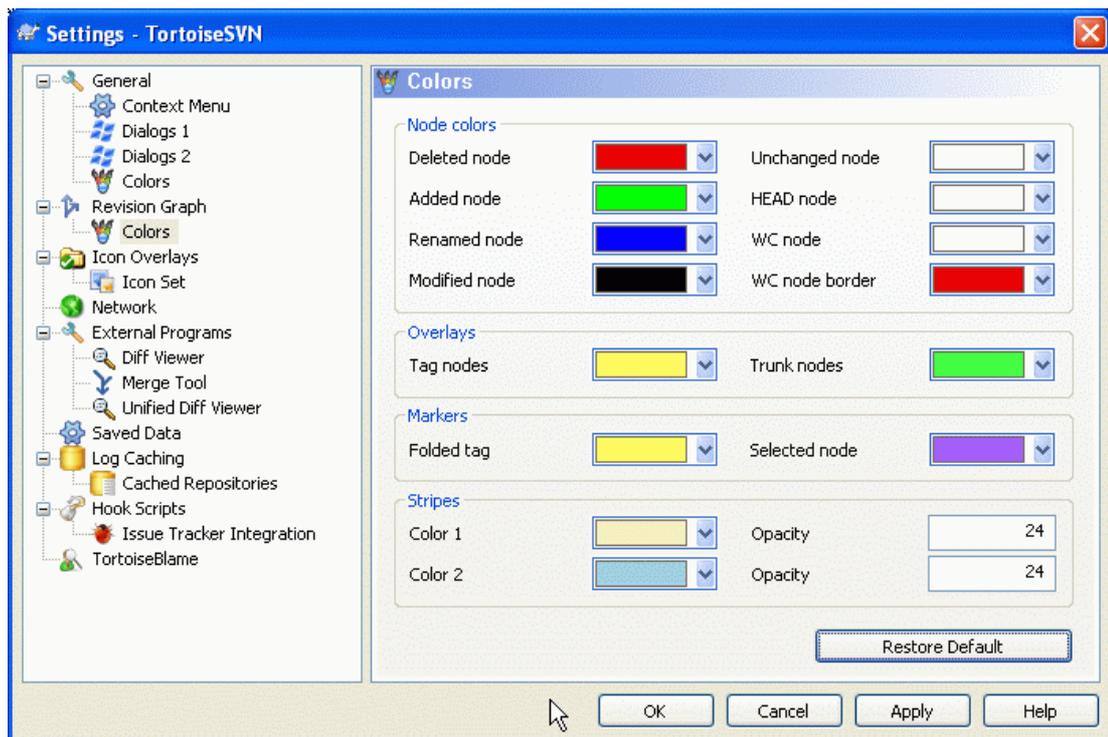


Abbildung 4.57. Der Einstellungsdialog, Farben des Revisionsgraphen

Auf dieser Seite legen sie die zu verwendenden Farben fest. Beachten Sie bitte das es sich hierbei um die Grundfarben handelt. Die meisten Knoten werden durch eine Überblendung der Farben für Knotentyp, Hintergrund und optional der Farbe für die Klassifizierung gekennzeichnet.

Gelöschter Knoten

Objekte, die in dieser Revision gelöscht und an keine andere Stelle verschoben wurden.

Hinzugefügter Knoten

Neu hinzugefügte oder kopierte (mit Historie hinzugefügte) Objekte.

Umbenannter Knoten

Objekte, die in dieser Revision an einer Stelle gelöscht und an einer anderen Stelle hinzugefügt wurden.

Veränderter Knoten

Einfache Veränderung in der nichts hinzugefügt oder gelöscht wurde.

Unveränderter Knoten

Kann verwendet werden, um die Revision, die als Quelle einer Kopie diente hervorzuheben, selbst wenn sich an dem Objekt für das der Revisionsgraph gestartet wurde, in dieser Revision nichts geändert hat.

HEAD Knoten

Aktuelle HEAD Revision im Projektarchiv.

Arbeitskopie

Falls Sie einen Extraknoten für Ihre geänderte Arbeitskopie anzeigen lassen, wird diese Farbe für die Fläche des Knotens verwendet.

Randfarbe für AK

Wenn Sie sich anzeigen lassen, ob Ihre Arbeitskopie verändert ist, wird diese Farbe für den Rand des Knotens verwendet, falls die Arbeitskopie verändert wurde.

Marken

Knoten, die als Marken klassifiziert wurden, können mit dieser Farbe überlagert werden.

Stamm

Knoten, die als Stamm klassifiziert wurden, können mit dieser Farbe überlagert werden.

Zusammengeklappte Marken

Falls Sie Marken zusammenklappen, um Platz zu sparen, werden sie an der Quelle der Kopie durch einen Block in der gewählten Farbe gekennzeichnet.

Selektierte Marken

Wenn Sie auf einen Knoten links-Klicken um diesen zu selektieren, wird der Selektionsmarker in dieser Farbe angezeigt.

Streifen

Diese Farben werden benutzt wenn der Graph in Unterbäume aufgeteilt und der Hintergrund in abwechselnden Streifen gezeichnet wird um die einzelnen Bäume besser unterscheiden zu können.

4.30.3. Überlagerte Symbole

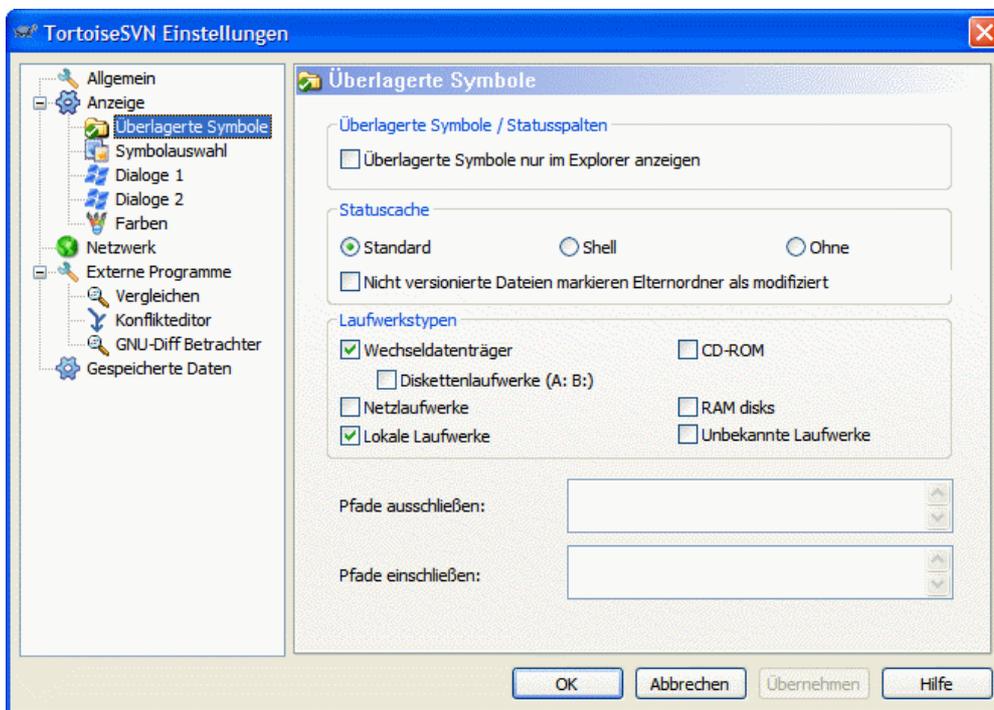


Abbildung 4.58. Der Einstellungsdialog, Symbolauswahl

In diesem Dialog können Sie die Objekte auswählen, für die TortoiseSVN überlagerte Symbole anzeigen soll.

Standardmäßig werden die überlagerten Symbole in allen Öffnen/Speichern-Dialogen und im Windows Explorer angezeigt. Wenn Sie wollen, dass die Symbole *nur* im Explorer angezeigt werden, wählen Sie die Option Überlagerte Symbole nur im Explorer anzeigen.

Ignorierte und nicht versionierte Objekte erhalten normalerweise kein überlagertes Symbol. Falls Sie diese Objekte in Arbeitskopien kennzeichnen wollen, aktivieren Sie die entsprechenden Optionen.

Sie können auch festlegen, dass Ordner als modifiziert angezeigt werden sollen, wenn Sie unversionierte Objekte enthalten. Das können Sie als Erinnerung nutzen, dass Sie neue, noch nicht versionierte, Dateien

angelegt haben. Diese Option ist nur verfügbar, wenn Sie die *Standard* Puffer Einstellungen nutzen (siehe unten).

Da es eine Weile dauern kann, den Status einer Arbeitskopie zu bestimmen, verwendet TortoiseSVN einen Puffer, um den Status zwischenspeichern, damit der Windows Explorer beim Anzeigen der überlagerten Symbole nicht so stark gebremst wird. Sie können hier festlegen, welche Art von Puffer TortoiseSVN benutzen soll:

Standard

Speichert sämtliche Statusinformation in einem separaten Prozess (*TSVNCache.exe*). Dieser Prozess beobachtet alle Laufwerke auf Änderungen und ermittelt den Status erneut, sobald sich eine Datei innerhalb einer Arbeitskopie ändert. Der Prozess läuft mit der niedrigsten Priorität, damit andere Programme nicht gebremst werden. Das bedeutet allerdings auch, dass die Statusinformation *nicht in Echtzeit* angezeigt wird, sondern dass es ein paar Sekunden dauern kann, bis sich die überlagerten Symbole ändern.

Vorteil: Die überlagerten Symbole zeigen den Status rekursiv, das bedeutet dass wenn sich eine Datei tief innerhalb der Arbeitskopie ändert, alle Ordner bis zur Basis der Arbeitskopie ebenfalls das „modifiziert“ Symbol anzeigen. Da der Prozess auch Benachrichtigungen an die Shell schicken kann, ändern sich die Symbole in der linken Baumansicht normalerweise auch entsprechend.

Nachteil: Der Prozess läuft permanent, auch wenn Sie nicht an Ihren Projekten arbeiten. Er benötigt außerdem, abhängig von Anzahl und Größe Ihrer Arbeitskopien 10-50 MB Hauptspeicher.

Shell

Die Daten werden direkt in der Shell Erweiterungs-DLL zwischengespeichert, aber nur für den aktuellen Ordner. Jedes Mal wenn Sie einen anderen Ordner öffnen, wird die Statusinformation erneut geladen.

Vorteil: benötigt nur sehr wenig Hauptspeicher (zirka 1 MB) und kann den Status in *Echtzeit* anzeigen.

Nachteil: Da nur ein Ordner zwischengespeichert wird, kann der Status nicht rekursiv angezeigt werden. Bei großen Arbeitskopien kann es länger als mit dem Standardpuffer dauern, einen Ordner anzuzeigen. Obendrein steht die MIME-Typ Spalte nicht zur Verfügung.

Ohne

Mit dieser Einstellung ermittelt TortoiseSVN keinerlei Statusinformationen im Explorer. Deshalb erhalten Dateien kein überlagertes Symbol und Ordner, wenn Sie versioniert sind, nur das 'Normal' Symbol. Es werden keine anderen Symbole überlagert und es stehen keine zusätzlichen Spalten zur Verfügung.

Vorteil: Benötigt absolut keinen zusätzlichen Speicher und verlangsamt den Explorer nicht.

Nachteil: Im Explorer steht keinerlei Statusinformation zur Verfügung. Sie müssen deshalb mit dem **Auf Änderungen prüfen**-Dialog nachschauen, welche Dateien in Ihrer Arbeitskopie verändert wurden.

Auf dieser Seite können Sie einstellen, für welche Laufwerkstypen überlagerte Symbole angezeigt werden. Standardmäßig sind nur Festplatten ausgewählt. Sie können auch alle überlagerten Symbole deaktivieren, aber wo liegt der Spaß darin?

Der Zugriff auf Netzwerklaufwerke kann sehr langsam sein, weshalb standardmäßig keine überlagerten Symbole für Netzwerklaufwerke angezeigt werden.

USB Flashlaufwerke stellen insofern einen Spezialfall dar, als das der Laufwerkstyp durch das Gerät selbst festgelegt wird. Manche erscheinen als Festplatten, andere wiederum als Wechselplatten.

Die **Ausschlusspfade** teilen TortoiseSVN mit, für welche Pfade die überlagerten Symbole *nicht* gezeichnet werden sollen. Dies ist nützlich, wenn Sie zum Beispiel sehr große Arbeitskopien haben,

welche große externe Bibliotheken, die Sie selbst nie ändern werden, enthalten. Sie können dann die entsprechenden Pfade gezielt ausschließen. Zum Beispiel:

`f:\development\SVN\Subversion` deaktiviert die überlagerten Symbole *nur* für diesen speziellen Ordner. In den darunter liegenden Ordnern werden Sie weiterhin alle Symbole sehen.

`f:\development\SVN\Subversion*` deaktiviert die überlagerten Symbole für *alle* Dateien und Ordner, deren Pfad mit `f:\development\SVN\Subversion` beginnt. Das bedeutet dass auch für alle Dateien und Ordner innerhalb keine überlagerten Symbole angezeigt werden.

Das gleiche gilt für die **Einschlusspfade**. Nur dass für diese Pfade die Symbole auch angezeigt werden, wenn sie zunächst für einen Laufwerks-Typ oder durch einen Ausschlusspfad deaktiviert wurden.

Anwender fragen manchmal, wie diese Einstellungen zusammenhängen. Die Antwort lautet:

```
WENN (der Pfad eingeschlossen ist)
    Zeige Symbole
WENN (der Pfad ein erlaubter Laufwerkstyp ist)
    UND (der Pfad nicht ausgeschlossen ist)
        Zeige Symbole
```

Für eingeschlossene Pfade werden die überlagerten Symbole *immer* angezeigt. Andernfalls werden die Symbole für die gewünschten Laufwerkstypen angezeigt *es sei denn* der Pfad ist ausgeschlossen.

TSVNCache.exe nutzt diese Pfade, um seine Aktivitäten einzuschränken. Wenn Sie wollen, dass der Puffer nur bestimmte Ordner überwacht, schalten Sie alle Laufwerkstypen aus und schließen nur die Ordner ein, die überwacht werden sollen.



SUBST Laufwerke ausschließen

Häufig ist es praktisch, zum Zugriff auf Ihre Arbeitskopien ein SUBST Laufwerk zu definieren, z.B. mit dem Befehl

```
subst T: C:\TortoiseSVN\trunk\doc
```

Das kann jedoch dazu führen, dass die überlagerten Symbole nicht aktualisiert werden, da der TSVNCache nur eine Benachrichtigung erhält, wenn sich eine Datei ändert und diese Benachrichtigung erfolgt normalerweise nur für den Originalpfad. Das bedeutet, dass die überlagerten Symbole auf dem SUBST Pfad unter Umständen nie aktualisiert werden.

Ein einfacher Weg, das Problem zu umgehen, ist den Originalpfad von der Anzeige der überlagerten Symbole auszuschließen, so dass die Symbole stattdessen auf dem `subst` Pfad erscheinen.

Manchmal möchten Sie vielleicht Ordner, die Arbeitskopien enthalten von TSVNCache ausschließen aber weiterhin einen optischen Hinweis darauf haben, dass es sich um versionierte Ordner handelt. Dazu wählen Sie die **Ausgeschlossene Ordner 'normal' anzeigen** Option. Damit werden versionierte, ausgeschlossene Ordner (Laufwerkstyp nicht aktiviert oder gezielt ausgeschlossen) mit dem Symbol für normal und aktuell angezeigt. Dies erinnert Sie daran, dass Sie eine Arbeitskopie vor sich haben, auch wenn die überlagerten Symbole nicht unbedingt korrekt sind. Dateien in solchen Ordnern erhalten kein überlagertes Symbol. Die Kontextmenüs sind weiterhin funktionsfähig, auch wenn keine Symbole überlagert werden.

Als Ausnahme davon werden die Laufwerke `A:` und `B:` niemals bei der **Ausgeschlossene Ordner 'normal' anzeigen** Option berücksichtigt. Der Grund dafür ist, dass Windows gezwungen ist, auf diese Laufwerke zuzugreifen, was selbst dann für eine Verzögerung von mehreren Sekunden sorgen kann, wenn Ihr PC mit einem Diskettenlaufwerk ausgestattet ist.

4.30.3.1. Auswahl der überlagerten Symbole

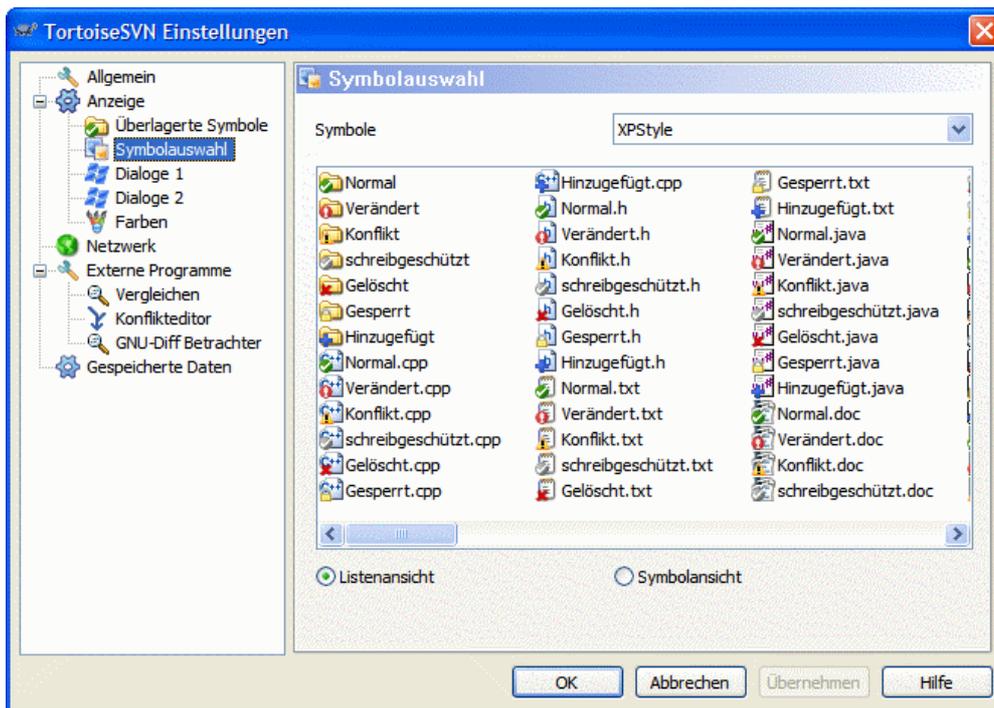


Abbildung 4.59. Der Einstellungsdialog, Symbolauswahl

Sie können in TortoiseSVN die überlagerten Symbole auswählen, die Ihnen am besten gefallen. Beachten Sie dass wenn Sie diese Einstellung ändern, der Symbolsatz erst nach einem Neustart des Computers vom Explorer übernommen wird.

4.30.4. Netzwerk Einstellungen

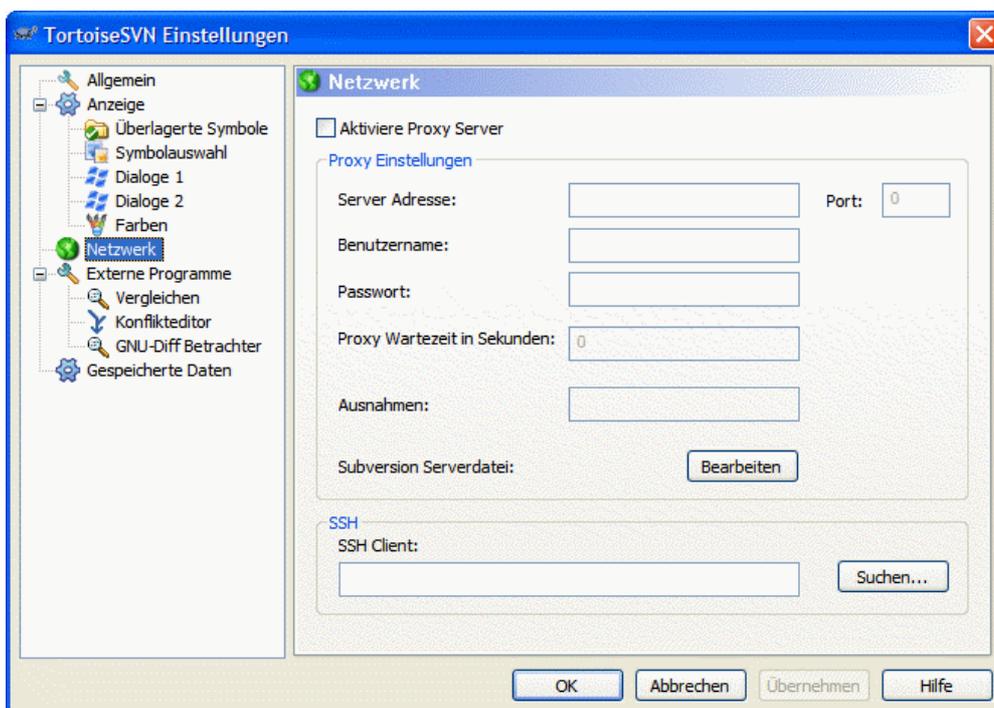


Abbildung 4.60. Der Einstellungsdialog, Netzwerkseite

Hier können Sie Einstellungen für einen Proxyserver vornehmen, sofern Sie einen solchen benötigen (z.B. bei einer Firmen-Firewall).

Wenn Sie Proxy Einstellungen für jedes Projektarchiv vornehmen wollen, müssen Sie das in der Subversion `servers` Datei einstellen. Über die **Bearbeiten** Schaltfläche kommen Sie direkt dort hin. Lesen Sie dazu im Kapitel *Runtime Configuration Area* [<http://svnbook.red-bean.com/en/1.5/svn.advanced.confarea.html>] des Subversion Buchs nach, wie die Datei aufgebaut ist.

Sie können hier auch angeben, welches Programm TortoiseSVN benutzen soll, um eine sichere Verbindung zu einem `svn+ssh` Projektarchiv herzustellen. Wir empfehlen Ihnen, dass Sie dazu TortoisePlink.exe verwenden. Dabei handelt es sich um eine für TortoiseSVN angepasste Version des bekannten Plink, die mit TortoiseSVN zusammen installiert wird. Damit nicht bei jeder Anmeldung eine DOS-Box erscheint, wurde TortoisePlink als fensterloses Programm übersetzt.

Sie müssen den vollständigen Pfad zu der ausführbaren Datei angeben. Für TortoisePlink.exe ist es das standard TortoiseSVN `bin` Verzeichnis. Benutzen Sie die **Suchen** Schaltfläche, um die Datei zu finden. Beachten Sie bitten, dass der Pfad in Anführungszeichen gesetzt werden muss, falls er Leerzeichen enthält, z.B.

```
"C:\Program Files\TortoiseSVN\bin\TortoisePlink.exe"
```

Ein Seiteneffekt davon ist, dass TortoisePlink ohne Fenster keine Möglichkeit hat, Fehlermeldungen anzuzeigen. Wenn die Anmeldung fehlschlägt, erscheint einfach `Unable to write to standard output`. Aus diesem Grund empfehlen wir, dass Sie die Konfiguration erst mit dem Standard Plink einrichten. Wenn alles funktioniert wechseln Sie zu TortoisePlink und verwenden die gleichen Parameter.

Für TortoisePlink steht keine eigene Dokumentation zur Verfügung, weil es sich nur um eine leichte Abwandlung von Plink handelt. Die Verwendung von Plink ist auf der *PuTTY Webseite* [<http://www.chiark.greenend.org.uk/~sgtatham/putty/>] detailliert beschrieben.

Um zu vermeiden, dass Sie wiederholt nach einem Passwort gefragt werden, können Sie ein Programm zum Zwischenspeichern des Passworts, wie Pageant einsetzen. Diese Anwendung steht ebenfalls auf der PuTTY Webseite zur Verfügung.

Abschließend bleibt festzuhalten, dass die Einrichtung von SSH auf Server und Client ein nicht-trivialer Prozess ist, der den Rahmen dieses Handbuchs sprengt. Sie finden eine weitere Anleitung unter *Subversion/TortoiseSVN SSH How-To* [http://tortoisesvn.net/ssh_howto].

4.30.5. Einstellungen für externe Programme

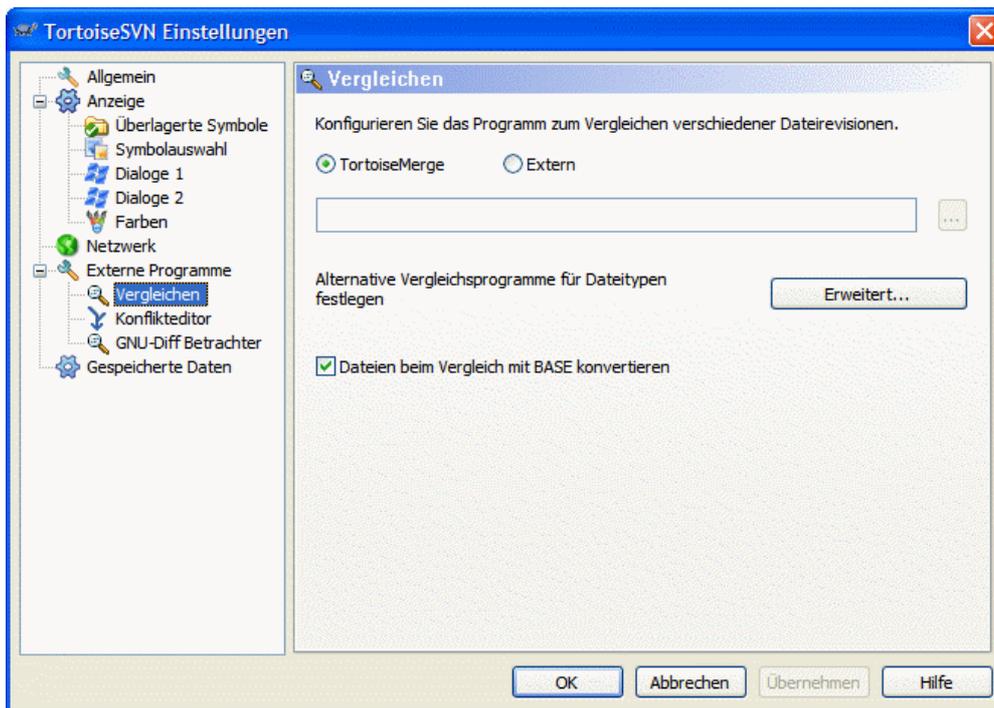


Abbildung 4.61. Der Einstellungsdialog, Externe Programme

Hier können Sie Ihre eigenen Vergleichs- und Konflikteditoren definieren, die TortoiseSVN benutzen soll. In der Standardeinstellung wird das mitgelieferte TortoiseMerge benutzt.

Lesen Sie in [Abschnitt 4.10.5, „Externe Programme“](#) nach, welche externen Vergleichs- und Konflikteditoren mit TortoiseSVN genutzt werden können.

4.30.5.1. Vergleichsprogramm

Ein externes Vergleichsprogramm, mit dem Sie verschiedene Revisionen einer Datei vergleichen können. Die Dateinamen müssen gemeinsam mit anderen Parametern per Kommandozeile übergeben werden. TortoiseSVN verwendet dazu Platzhalter, denen ein % vorangestellt wird. Die Reihenfolge der Parameter auf der Kommandozeile hängt von dem von Ihnen verwendeten Vergleichsprogramm ab.

%base

Die Originaldatei ohne Ihre Änderungen

%bname

Der Fenstertitel für die Originaldatei

%mine

Die Datei mit Ihren eigenen Änderungen

%yname

Der Fenstertitel für Ihre Datei

Bei den Fenstertiteln handelt es sich nicht um reine Dateinamen. TortoiseSVN behandelt sie als Anzeigenamen und stellt den Titel entsprechend zusammen. Wenn Sie z.B. ein Vergleich zwischen einer Datei in Revision 123 und der Arbeitskopie machen, werden die Titel `Dateiname : Revision 123` und `Dateiname : Arbeitskopie` sein.

Zum Beispiel für ExamDiff Pro:

```
C:\Pfad-Zu\ExamDiff.exe %base %mine --left_display_name:%bname
--right_display_name:%yname
```

oder für KDiff3:

```
C:\Pfad-Zu\Kdiff3.exe %base %mine --L1 %bname --L2 %yname
```

oder mit WinMerge:

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

oder mit Araxis:

```
C:\Pfad-Zu\compare.exe /max /wait /title1:%bname /title2:%yname
%base %mine
```

Wenn Sie die `svn:keywords` Eigenschaft gesetzt haben, um Schlüsselwörter, insbesondere die *Revision* einer Datei, zu expandieren, kann ein Unterschied zwischen Dateien entstehen, der nur aus dem aktuellen Wert des Schlüsselwortes besteht. Ebenso wird, wenn Sie `svn:eol-style = native` setzen, die BASE Datei reine LF Zeilenenden haben, während Ihre Datei CR-LF Zeilenenden besitzt. TortoiseSVN wird diese Unterschiede normalerweise verbergen, indem es zuerst die Schlüsselwörter und Zeilenenden BASE in der Datei erweitert, bevor die Vergleichsoperation durchgeführt wird. Dies kann bei großen Dateien viel Zeit beanspruchen. Wenn die Option *Dateien beim Vergleich mit BASE konvertieren* nicht gewählt ist, wird TortoiseSVN diese Vorbereitungsarbeiten nicht durchführen und damit Zeit sparen.

Sie können auch ein eigenes Programm für Subversion Eigenschaften angeben. Da es sich dabei um einfache, kurze Texte handelt, wollen Sie vielleicht einen einfachen, kompakten Betrachter verwenden

Wenn Sie ein alternatives Vergleichsprogramm eingerichtet haben, können Sie TortoiseMerge *und* das andere Programm aus den Kontextmenüs heraus aufrufen. *Kontextmenü* → *Vergleich* ruft das primäre Vergleichsprogramm und *Umsch+Kontextmenü* → *Vergleich* das sekundäre Vergleichsprogramm auf.

4.30.5.2. Konflikteditor

Ein externes Programm, mit dem Sie Konflikte in Dateien auflösen können. Die Parameterübergabe an das Programm erfolgt genau so wie beim Vergleichsprogramm.

`%base`

Die Originaldatei ohne irgendwelche Änderungen

`%bname`

Der Fenstertitel für die Originaldatei

`%mine`

Die Datei mit Ihren eigenen Änderungen

`%yname`

Der Fenstertitel für Ihre Datei

`%theirs`

Die Datei mit den letzten Änderungen im Projektarchiv

`%tname`

Der Fenstertitel für die Datei im Projektarchiv

`%merged`

Die Konfliktdatei (das Ergebnis der Zusammenführen Operation)

`%mname`

Der Fenstertitel für die Ergebnisdatei

Zum Beispiel für Perforce Merge:

```
C:\Pfad-Zu\P4Merge.exe %base %theirs %mine %merged
```

oder für KDiff3:

```
C:\Pfad-Zu\Kdiff3.exe %base %mine %theirs -o %merged
--L1 %bname --L2 %ynname --L3 %tname
```

oder mit Araxis:

```
C:\Pfad-Zu\compare.exe /max /wait /3 /title1:%tname /title2:%bname
/title3:%ynname %theirs %base %mine %merged /a2
```

oder mit WinMerge (2.8 oder neuer):

```
C:\Path-Zu\WinMerge.exe %merged
```

4.30.5.3. Erweiterte Einstellungen für Vergleichs- und Konflikteditor

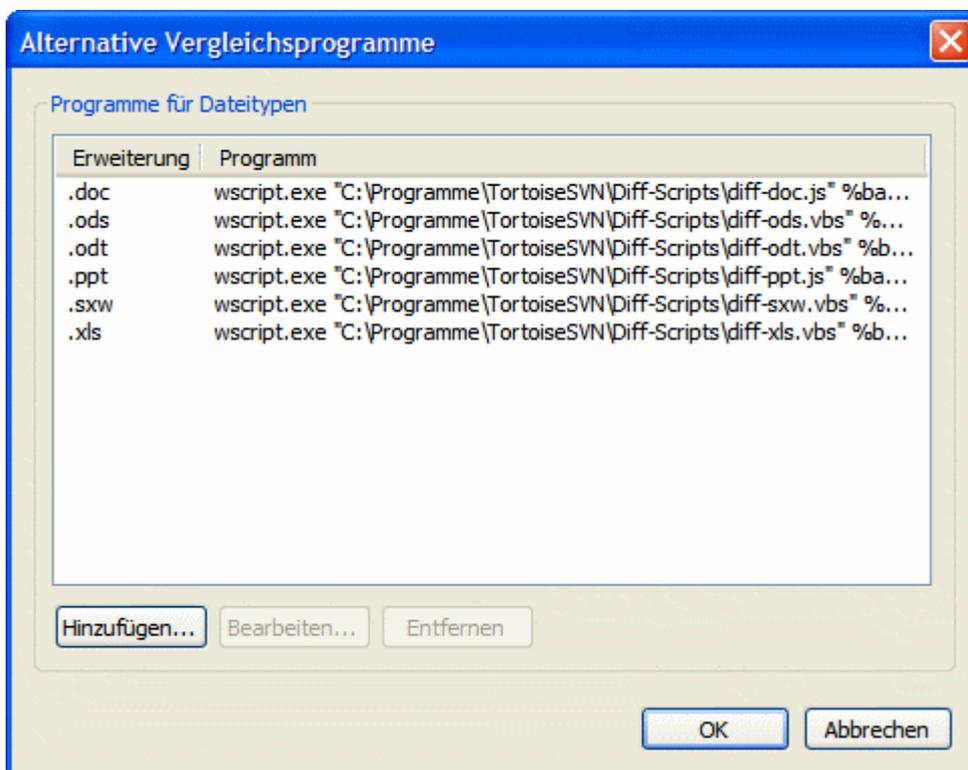


Abbildung 4.62. Erweiterte Einstellungen für Vergleichs- und Konflikteditor

In den erweiterten Einstellungen können Sie für jede Dateiendung ein eigenes Vergleichsprogramm oder Konflikteditor angeben, wenn Sie möchten. So können Sie z.B. Photoshop als „Vergleichsprogramm“ für .jpg Dateien festlegen :-). Sie können auch allgemein einen `svn:mime-type` mit einem Vergleichsprogramm oder Konflikteditor verknüpfen.

Um Dateiendungen mit Anwendungen zu verknüpfen, geben Sie die Erweiterung an, z.B. *.bmp für Windows Bitmap Dateien. Um eine Verknüpfung über die `svn:mime-type` Eigenschaft herzustellen, geben Sie den Mime-Typ, inklusive Schrägstrich an, also `text/xml`.

4.30.5.4. Standard-Diff Betrachter

Ein Programm um GNU-Diff Dateien (Patch Dateien) zu betrachten. Sie brauchen keine Parameter anzugeben. Der Standard ist, erst nach einem Betrachter für .diff und dann für .txt Dateien zu

suchen. Wenn Sie keinen Betrachter für `.diff` Dateien eingerichtet haben, wird normalerweise Notepad aufgerufen.

Das Original Windows Notepad Programm zeigt Zeilenenden, die nicht dem Windows Standard (CR-LF) entsprechen, nicht korrekt an. Da die meisten GNU Diffdateien Unix Zeilenenden (LF) besitzen ist das Resultat entsprechend. Sie können sich jedoch einen kostenlosen Notepad Ersatz *Notepad2* [<http://www.flos-freeware.ch/notepad2.html>] herunterladen, der nicht nur die Zeilenenden korrekt anzeigt, sondern auch die hinzugefügten und gelöschten Zeilen farblich hervorhebt.

4.30.6. Gespeicherte Daten

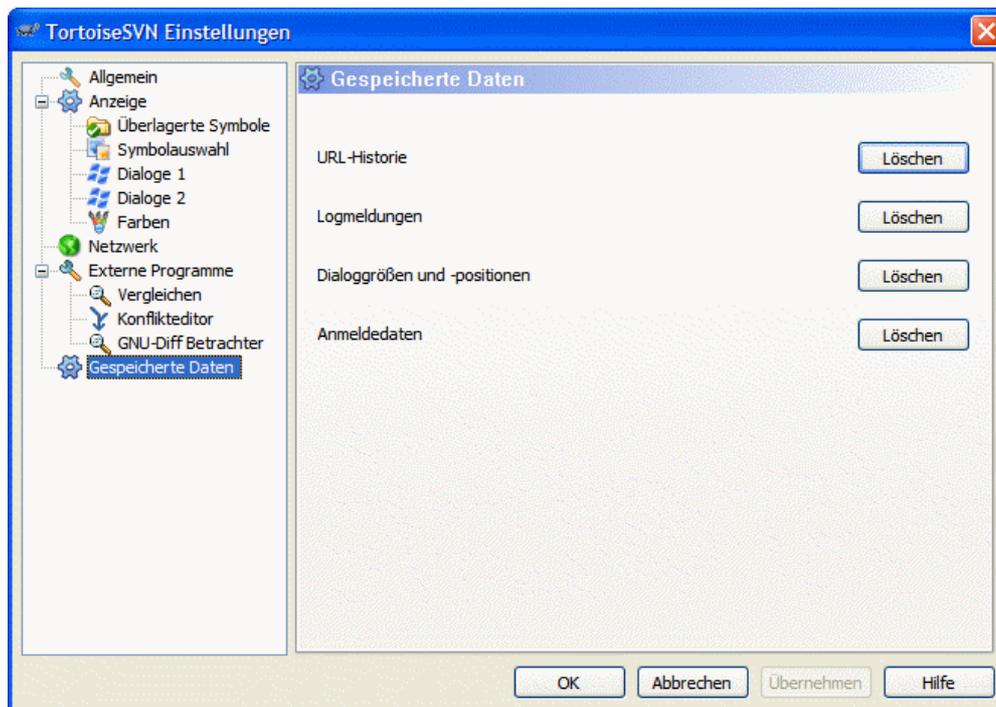


Abbildung 4.63. Der Einstellungsdialog, gespeicherte Daten

Zu Ihrer Bequemlichkeit speichert TortoiseSVN viele Ihrer Einstellungen und merkt sich, welche Sie zuletzt verwendet haben. Auf dieser Seite können Sie die Informationen wieder löschen.

URL-Historie

Jedes Mal, wenn Sie eine Arbeitskopie auschecken, Änderungen zusammenführen oder ein Projektarchiv betrachten, merkt sich TortoiseSVN die entsprechende URL und bietet Sie in einer Liste der zuletzt benutzten URLs an. Über die Zeit kann sich diese Liste mit veralteten URLs füllen, so dass es sinnvoll sein kann sie periodisch zu löschen.

Wenn Sie einen einzelnen Eintrag aus eine Kombinationsliste löschen wollen, können Sie das direkt in der Liste tun. Klicken Sie auf den Pfeil, um die Kombinationsliste zu öffnen, fahren Sie mit der Maus auf den zu löschenden Eintrag und drücken Sie **Umsch+Löschen**.

Logmeldungen (Eingabedialog)

TortoiseSVN speichert die letzten von Ihnen eingegebenen Logmeldungen. Diese werden für jedes Projektarchiv einzeln abgespeichert, so dass die Liste, wenn Sie viele Projektarchive nutzen, recht groß werden kann.

Logmeldungen (Zeige Log-Dialog)

TortoiseSVN speichert bereits übertragene Logmeldungen lokal, um beim nächsten Anzeigen desselben Logs Zeit zu sparen. Wenn jemand eine Logmeldung ändert, die bereits lokal vorhanden ist, werden Sie die Änderung nicht sehen bevor Sie den Zwischenspeicher gelöscht haben. Das

Speichern von Logmeldungen wird auf Log-Puffer Seite in den TortoiseSVN Einstellungen aktiviert.

Dialoggrößen und -positionen

Viele Dialoge merken sich die zuletzt verwendete Größe und Position auf dem Bildschirm.

Anmeldedaten

Wenn Sie sich an einem Subversion Server anmelden, können Anmeldenname und Passwort lokal gespeichert werden, damit Sie sie nicht ständig eingeben müssen. Vielleicht möchten Sie die Anmeldedaten aus Sicherheitsgründen löschen oder weil Sie unter einem anderen Namen auf das Projektarchiv zugreifen wollen ... Weiß John, dass Sie seinen PC benutzen?

Wie Sie die Anmeldedaten für einen bestimmten Server löschen, können Sie in [Abschnitt 4.1.5, „Anmeldung“](#) nachlesen.

Aktionslog

TortoiseSVN protokolliert alles mit, was in seine Fortschrittsdialoge geschrieben wird. In diesem Protokoll können Sie zum Beispiel nachschauen, was bei einer kurz zuvor durchgeführten Aktualisierung geschah.

Die Länge der Protokolldatei ist begrenzt und älterer Inhalt wird verworfen, sobald die Datei zu groß wird. Standardmäßig werden 4000 Zeilen zwischengespeichert, aber Sie können diese Vorgabe ändern

Von hier aus können Sie die Logdatei anschauen oder auch löschen.

4.30.7. Log-Puffer

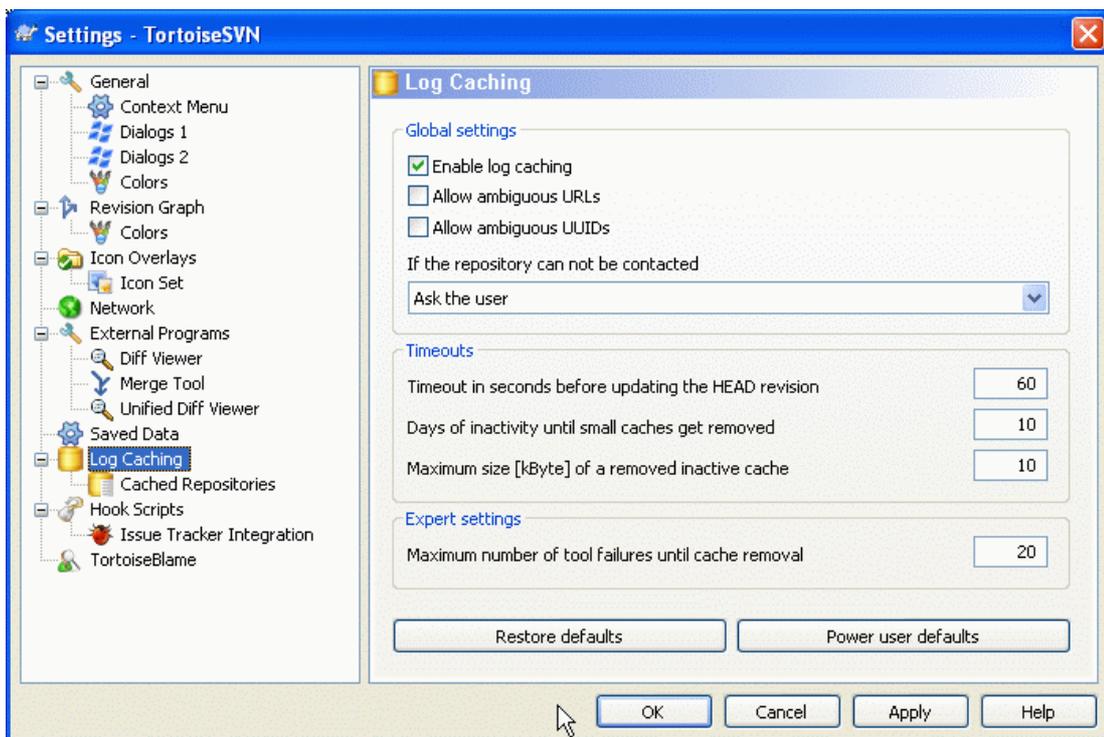


Abbildung 4.64. Der Einstellungsdialog, Log-Puffer

Dieser Dialog ermöglicht es Ihnen, den Log-Puffer von TortoiseSVN einzurichten. Dieser hält eine lokale Kopie von Logmeldungen und geänderten Pfaden vor, um zeitraubende Zugriffe auf den Server zu sparen. Die Verwendung des Log-Puffers kann die Anzeige des Log-Dialogs und des Revisionsgraphen drastisch beschleunigen. Ein weitere Vorteil ist, dass auch ohne Kontakt zum Server auf die Logmeldungen zugegriffen werden kann.

Log-Puffer aktivieren

Aktiviert den Log-Puffer sobald Logdaten angefragt werden. Falls aktiv, werden Daten, sofern verfügbar, aus dem Puffer gelesen und nicht im Puffer vorhandene Meldungen werden vom Server geholt und im Puffer abgelegt.

Wenn der Puffer nicht aktiviert ist, werden die Daten stets vom Server geholt und nicht lokal gespeichert.

Mehrdeutige URLs zulassen

Für den Fall dass die zu einem Server Verbindung aufnehmen welcher dieselbe URL für alle Projektarchive benutzt aktivieren Sie diese Checkbox. Zum Beispiel verwenden ältere Versionen von `svnbridge` immer dieselbe URL. Falls Sie dies nicht benötigen, lassen Sie die Checkbox inaktiv um die Performance zu verbessern.

Mehrdeutige UUIDs zulassen

Einige Subversion Hosts vergeben für alle Projektarchiven dieselbe UUID. Sie haben dies vielleicht durch einfaches Kopieren eines Projektarchivs selbst getan, anstatt dass Sie ein neues erstellt haben. Aus verschiedenen Gründen ist dies sehr schlecht - eine UUID sollte *einmalig* sein. Trotz all dem kann der Log-Puffer in solchen Situationen dennoch funktionieren wenn sie diese Option aktivieren. Wenn Sie dies nicht benötigen, lassen Sie die Option inaktiv um die Performance zu verbessern.

Falls das Projektarchiv nicht erreichbar ist

Wenn Sie keine Verbindung zum Netz haben oder der Server nicht verfügbar ist, kann mit Hilfe des Log-Puffers weiterhin auf die Logmeldungen zugegriffen werden. Natürlich ist es möglich, dass diese Daten nicht mehr aktuell sind. Aus diesem Grund können Sie mit Optionen festlegen, ob diese Funktion genutzt werden soll.

Wenn Logdaten ohne Zugriff auf den Server aus dem Puffer genommen werden, zeigt der Dialog, der diese Daten verwendet den *Offline* Status in der Titelzeile an.

Wartezeit vor Aktualisierung der HEAD Revision

Wenn Sie den Log-Dialog aufrufen, soll normalerweise der Server kontaktiert werden, um neue Logmeldungen abzurufen. Wenn die eingestellte Wartezeit nicht Null ist, wird der Server nur kontaktiert, falls die Wartezeit seit dem letzten Aufruf abgelaufen ist. Dies kann die Reaktionszeit beim Aufruf des Log-Dialogs beschleunigen, wenn Sie diesen häufig aufrufen und der Server langsam ist. Andererseits ist es möglich, dass die Logdaten dadurch nicht aktuell sind. Wenn Sie diese Funktion nutzen wollen, würden wir einen Wert von 300 Sekunden (5 Minuten) als Kompromiss vorschlagen.

Inaktive Tage, bis kleine Puffer gelöscht werden

Wenn Sie viele Projektarchive benutzen werden mit der Zeit viele Log-Puffer angesammelt. Wenn Sie diese nicht oft benutzen wird TortoiseSVN diese nach einer bestimmten Zeit löschen. Hiermit konfigurieren Sie die Löschfunktion.

Maximale Größe eines gelöschten inaktiven Puffers

Größere Puffer benötigen länger beim erneuten Erstellen, deshalb löscht TortoiseSVN nur die kleineren Puffer. Sie können die Löschr-Grenze mit diesem Wert einstellen.

Maximale Anzahl von Fehlschlägen bis ein Puffer gelöscht wird

Manchmal kann etwas mit dem Puffern schiefgehen und verursacht einen Absturz. Wenn dies passiert wird der Puffer normalerweise automatisch gelöscht, um zu verhindern dass dasselbe Problem erneut auftritt. Wenn Sie den weniger stabilen nightly-build benutzen können Sie versuchen den Puffer trotzdem zu behalten.

4.30.7.1. Gepufferte Projektarchive

Auf der nächsten Seite sehen Sie eine Liste der Projektarchive, die lokal gepuffert werden sowie den vom Puffer belegten Speicherplatz. Wenn Sie eines der Projektarchive wählen, können Sie die weiteren Funktionen aufrufen.

Klicken Sie auf **Aktualisieren**, um den Puffer aufzufrischen und alle Löcher zu füllen. Für ein großes Projektarchiv kann das sehr viel Zeit in Anspruch nehmen. Wenn Sie nicht am Netz sind, steht Ihnen dafür der bestmögliche Puffer zur Verfügung.

Klicken Sie auf **Export**, um den gesamten Puffer in eine CSV Datei zu exportieren. Dies könnte Ihnen nützlich sein, wenn Sie die Logdaten in einem externen Programm weiter verarbeiten wollen, ist aber in erster Linie für die TortoiseSVN Entwickler gedacht.

Klicken Sie auf **Löschen**, um sämtliche gepufferten Daten für die gewählten Projektarchive zu löschen. Dadurch wird der Puffer für die Projektarchive nicht deaktiviert, so dass beim nächsten Zugriff auf das Log, der Puffer neu aufgebaut wird.

4.30.7.2. Log-Puffer Statistiken



Abbildung 4.65. Der Einstellungsdialog, Log-Puffer Statistiken

Klicken Sie auf **Details**, um detaillierte Statistiken für einen bestimmten Puffer anzusehen. Viele der dort angezeigten Werte sind vor allem für die Entwickler von TortoiseSVN interessant, so dass sie nicht alle im Detail angezeigt werden.

RAM

Der für diesen Puffer benötigte Hauptspeicher

Platte

Der für den Puffer benötigte Festplattenspeicher. Die Daten sind komprimiert, so dass sich der Verbrauch normalerweise in Grenzen hält.

Verbindung

Zeigt an, ob das Projektarchiv beim letzten Zugriff auf den Puffer verfügbar war.

Letzte Aktualisierung

Der Zeitpunkt an dem der Pufferinhalt zuletzt geändert wurde.

Letzte HEAD Aktualisierung

Das letzte Mal, dass eine HEAD Revision vom Server abgerufen wurde.

Autoren

Die Anzahl verschiedener Autoren mit Logmeldungen, die im Puffer gespeichert ist.

Pfade

Die Anzahl der aufgelisteten Pfade, wie man sie mit `svn log -v` sehen würde.

Lücken

Die Anzahl von Revisionsbereichen, welche noch nicht gepuffert sind, weil sie bisher nicht abgefragt wurden. Dies ist ein Maß für die Anzahl der Lücken im Puffer.

Max Revision

Die höchste, im Puffer gespeicherte, Revisionsnummer

Anzahl Revisionen

Die Anzahl der im Puffer gespeicherten Revisionen. Dies ist ein weiteres Maß für die Vollständigkeit des Puffers.

4.30.8. Clientseitige Aktionskripte

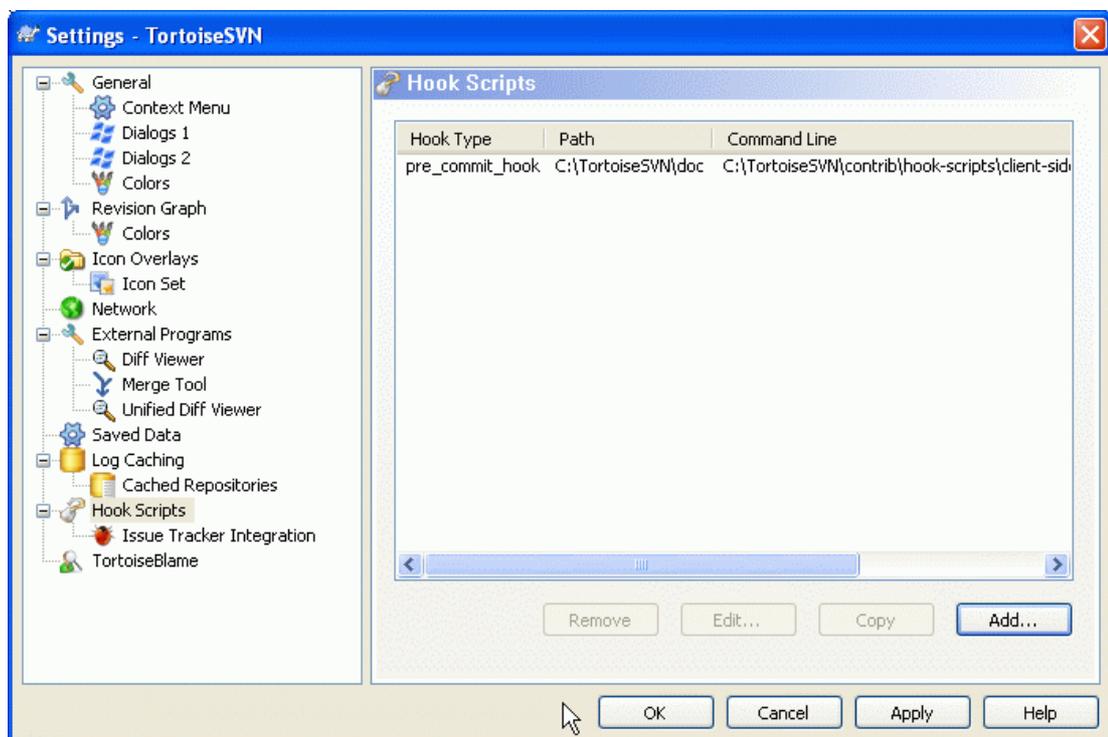


Abbildung 4.66. Der Einstellungsdialog, Aktionskripte

Dieser Dialog erlaubt Ihnen Aktionskripte festzulegen, die automatisch bei bestimmten Subversionaktionen ausgeführt werden. Im Gegensatz zu den in [Abschnitt 3.3](#), „Serverseitige Aktionskripte“ beschriebenen serverseitigen Aktionskripten, werden diese Skripte lokal ausgeführt.

Eine mögliche Anwendung für solche Aktionsskripte könnte z.B. sein, eine Anwendung wie SubWCRev.exe aufzurufen, um Versionsnummern nach einer Übertragung zu aktualisieren oder um ein Projekt neu zu erzeugen.

Aus verschiedenen implementierungs- und sicherheitstechnischen Gründen sind Aktionsskripte lokal auf einem Rechner und nicht als Eigenschaften des Projekts definiert. Sie legen damit fest, was passieren soll, unabhängig davon, was eine andere Person in das Projektarchiv überträgt. Natürlich können Sie jederzeit ein Skript aufrufen, das sich selbst unter Versionskontrolle befindet.

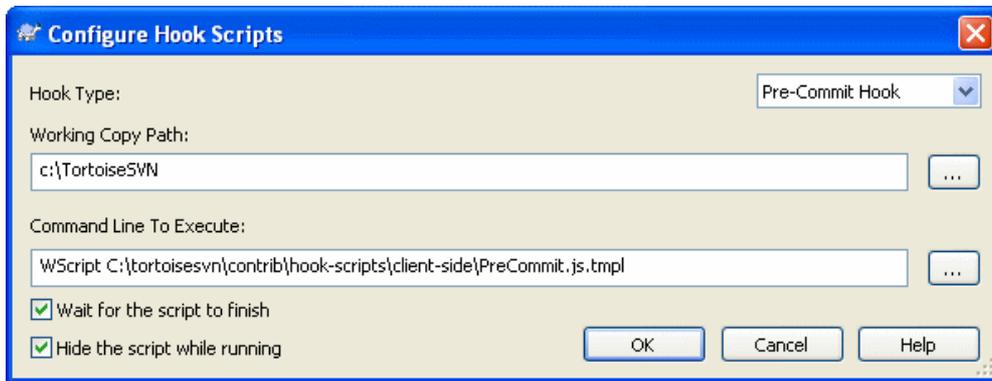


Abbildung 4.67. Der Einstellungsdialog, Aktionsskripte einrichten

Um ein neues Aktionsskript anzulegen, klicken Sie einfach auf die Hinzufügen Schaltfläche und füllen die Details aus.

Es stehen derzeit sechs Typen von Aktionsskripten zur Verfügung

Start Übertragung

Wird aufgerufen, bevor der Übertragen-Dialog angezeigt wird. Dieses Skript können Sie zum Beispiel benutzen, wenn sich durch die Aktion die Liste der zu übertragenden Dateien oder die Logmeldung ändert. Sie sollten sich dessen bewusst sein, dass dadurch, dass das Aktionsskript zu einem frühen Zeitpunkt aufgerufen wird, die vollständige Liste der zur Übertragung gewählten Objekte nicht zur Verfügung steht.

Vor Übertragung

Wird aufgerufen, nachdem der Anwender die OK Schaltfläche im Übertragen-Dialog geklickt hat und bevor die Übertragung selbst beginnt. Diesem Aktionsskript steht die vollständige Liste der zu übertragenden Objekte zur Verfügung.

Nach Übertragung

Wird nach dem Ende der Übertragung aufgerufen, unabhängig davon, ob die Übertragung erfolgreich war oder nicht.

Start Aktualisierung

Wird aufgerufen, bevor der „Aktualisiere zu Revision“ Dialog angezeigt wird.

Vor Aktualisierung

Wird aufgerufen, bevor die Aktualisierung beginnt.

Nach Aktualisierung

Wird nach dem Ende der Aktualisierung aufgerufen, unabhängig davon, ob die Aktualisierung erfolgreich war oder nicht.

Ein Aktionsskript wird für einen bestimmten Pfad einer Arbeitskopie definiert. Sie müssen nur den obersten Pfad angeben. Wenn Sie eine Aktion in einem Unterverzeichnis durchführen, wird TortoiseSVN automatisch weiter oben nach einem passenden Pfad suchen.

Als nächstes müssen Sie die auszuführende Befehlszeile angeben. Sie können eine beliebige ausführbare Datei (.bat, .exe, ...) mit einer gültigen Windows Dateiverknüpfung angeben also z.B. auch ein Perl-Skript.

Die Befehlszeile kann mehrere Parameter enthalten, die durch TortoiseSVN ausgefüllt werden. Welche Parameter zur Verfügung stehen, hängt vom aufgerufenen Aktionskript ab. Jedes Aktionskript hat seine eigenen Parameter, die in der folgenden Reihenfolge übergeben werden:

Start Übertragung

PATHMESSAGEFILECWD

Vor Übertragung

PATHDEPTHMESSAGEFILECWD

Nach Übertragung

PATHDEPTHMESSAGEFILEREVISIONERRORCWD

Start Aktualisierung

PATHCWD

Vor Aktualisierung

PATHDEPTHREVISIONCWD

Nach Aktualisierung

PATHDEPTHREVISIONERRORCWD

Die Bedeutung der Variablen wird in der folgenden Liste erklärt:

PATH

Ein Pfad zu einer Temporärdatei, der alle Pfade enthält für die die Aktion durchgeführt wurde. Jeder Pfad steht auf einer eigenen Zeile in der Temporärdatei.

DEPTH

Die Tiefe mit der die Übertragung/Aktualisierung durchgeführt wird.

Mögliche Werte sind:

-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

Pfad zu einer Datei, die die Logmeldung für die Übertragung im UTF-8 Format enthält. Nach erfolgreicher Ausführung der `start-commit` Aktion, wird die Logmeldung zurückgelesen, so dass die Aktion die Meldung zwischendurch modifizieren kann.

REVISION

Die Revisionsnummer des Projektarchivs nach einer abgeschlossenen Übertragung bzw. zu der das Projektarchive aktualisiert werden soll.

ERROR

Pfad zu einer Datei, die die Fehlermeldung enthält. Falls kein Fehler auftrat ist die Datei leer.

CWD

Das aktuelle Arbeitsverzeichnis, in dem das Skript ausgeführt wird. Es wird auf das gemeinsame Basisverzeichnis aller betroffenen Pfade gesetzt.

Beachten Sie bitte Folgendes: Obwohl wir zu Ihrer Bequemlichkeit die Parameter angeben, müssen Sie in den Einstellungen des Aktionsskripts nicht darauf verweisen. Alle, für ein bestimmtes Aktionsskript zur Verfügung stehenden Parameter werden immer an das Skript übergeben, ob Sie das wollen oder nicht ;-)

Falls Sie die Subversionoperation zurückhalten wollen, bis das Aktionsskript fertig ist, wählen Sie die Option **Warte auf Beendigung des Skripts**.

Normalerweise wollen Sie bestimmt die hässlichen DOS-Fenster verbergen während das Aktionsskript läuft, weshalb **Skript beim Ablauf verbergen** standardmäßig gewählt ist.

Beispielaktionsskripte finden sich im `contrib` Verzeichnis des *TortoiseSVN Projektarchivs* [<http://tortoisesvn.googlecode.com/svn/trunk/contrib/hook-scripts>] finden. (**Abschnitt 3, „TortoiseSVN ist frei!“** erklärt, wie man auf das TortoiseSVN Projektarchiv zugreift).

4.30.8.1. Integration mit Fehlerverfolgungssystemen

TortoiseSVN kann über eine COM Schnittstelle aus dem Übertragen-Dialog Einträge von einem Fehlerverfolgungssystem abfragen. Die Verwendung solcher Module wird in **Abschnitt 4.28.2, „Informationen vom Fehlerverfolgungssystem beziehen“** beschrieben. Falls Ihr Systemadministrator Ihnen ein solches Modul zur Verfügung gestellt hat, das Sie installiert und registriert haben, stellen Sie auf dieser Seite ein, wie es in Ihre Arbeitskopien integriert wird.

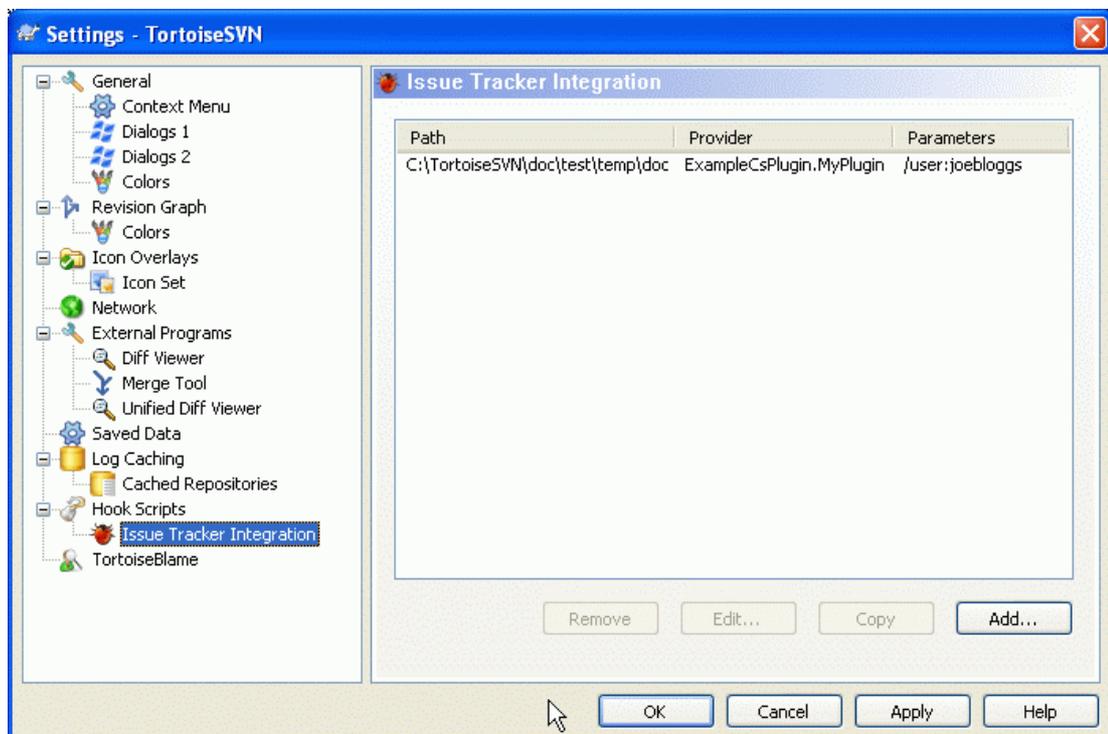


Abbildung 4.68. Der Einstellungsdialog, Integration eines Fehlerverfolgungssystems

Klicken Sie auf **Hinzufügen...**, um das Modul einer bestimmten Arbeitskopie zuzuordnen. Sie geben den Pfad zur Arbeitskopie an, wählen das gewünschte Modul aus einer List der zur Verfügung stehenden Module und geben die notwendigen Übergabeparameter an. Die Parameter sind spezifisch für das Modul, könnten aber zum Beispiel Ihren Benutzernamen im Fehlerverfolgungssystem beinhalten, so dass das Modul alle Ihnen zugeordneten Einträge ermitteln kann.

Wenn Sie möchten, dass alle Anwender dasselbe COM Modul für Ihr Fehlerverfolgungssystem verwenden, können Sie das Modul auch über die Eigenschaften `bugtraq:provideruuid` und `bugtraq:providerparams` einrichten.

`bugtraq:provideruuid`

Diese Eigenschaft enthält die COM UUID des `IBugtraqProvider`, zum Beispiel `{91974081-2DC7-4FB1-B3BE-0DE1C8D6CE4E}`. (Das Beispiel ist die UUID des [Gurtle bugtraq provider](http://code.google.com/p/gurtle/) [http://code.google.com/p/gurtle/], einer Anbindung für das [Google Code](http://code.google.com/hosting/) [http://code.google.com/hosting/] Fehlerverfolgungssystem).

`bugtraq:providerparams`

Diese Eigenschaft enthält die Parameter, die an den `IBugtraqProvider` übergeben werden.

Bitte lesen Sie in der Dokumentation zu Ihrem `IBugtraqProvider` Modul nach, was in diesen beiden Eigenschaften einzutragen ist.

4.30.9. TortoiseBlame Einstellungen

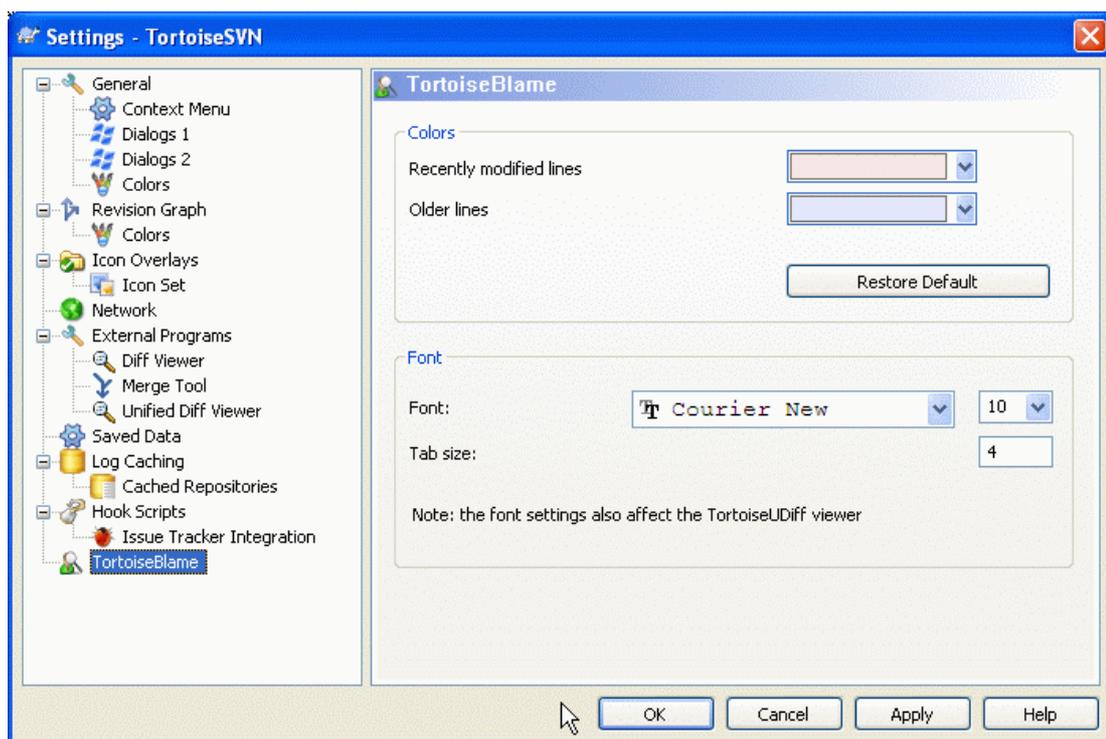


Abbildung 4.69. Der Einstellungsdialog, TortoiseBlame

Die Einstellungen von TortoiseBlame werden im TortoiseSVN Einstellungsdialog vorgenommen, nicht in TortoiseBlame selbst.

Farben

TortoiseBlame kann mit Hilfe der Hintergrundfarbe das Alter der Zeilen in einer Datei anzeigen. Sie können die Anfangs- und Endfarbe für die neueste bzw. älteste Revision festlegen. TortoiseBlame wird, entsprechend der Revisionsnummer, der Zeile eine Farbe zwischen diesen Werten zuordnen.

Schrift

Sie können die Schriftart und Größe zur Anzeige des Textes festlegen. Diese Einstellung wird sowohl für den Inhalt als auch für die Autor- und Revisionsinformationen auf der linken Seite verwendet.

Tabulator

Legt fest, wie viele Leerzeichen für einen Tabulator eingesetzt werden sollen.

4.30.10. Einstellungen in der Registrierung

Einige selten verwendete Einstellungen können nur über die Windows Registrierung verändert werden. Es muss wohl nicht extra erwähnt werden, dass Sie Registrierungseinträge nur verändern sollten, wenn Sie genau wissen, was Sie tun.

Konfiguration

Sie können einen anderen Ort für die Subversion Konfigurationsdateien angeben, indem Sie den Schlüssel `HKCU\Software\TortoiseSVN\ConfigDir` ändern. Dies beeinflusst alle TortoiseSVN Aktionen.

Statuszeilensymbol für Puffer

Um ein Puffer-Symbol zur Statusleiste hinzuzufügen, erzeugen Sie einen `DWORD` Schlüssel namens `HKCU\Software\TortoiseSVN\CacheTrayIcon` mit dem Wert 1. Diese Einstellung ist eigentlich nur für Entwickler gedacht, die den TortoiseSVN Puffer sanft beenden wollen.

Debug

Um die Befehlszeilenparameter anzuzeigen, welche von der Shell-Erweiterung an `TortoiseProc.exe` weitergegeben werden erstellen Sie einen `DWORD` Schlüssel mit Wert 1 unter `HKCU\Software\TortoiseSVN\Debug`.

Kontextmenüsymbbole

Dies kann nützlich sein wenn Sie einen anderen Dateimanager als den Windows Explorer benutzen oder wenn Sie Probleme mit der Darstellung der Kontextmenüs haben. Erstellen Sie einen `DWORD` Wert mit dem Wert 0 unter `HKCU\Software\TortoiseSVN>ShowContextMenuIcons`, wenn Sie möchten dass TortoiseSVN keine Icons im Kontextmenü anzeigen soll.

Verzögerte Symbolaktualisierung

Wenn Sie verhindern möchten, dass der Explorer die überlagerten Symbole aktualisiert, während ein TortoiseSVN Befehl läuft, erzeugen Sie einen `DWORD` Schlüssel namens `HKCU\Software\TortoiseSVN\BlockStatus` mit dem Wert 1.

URL für Aktualisierungsprüfung

`HKCU\Software\TortoiseSVN\UpdateCheckURL` enthält die URL von welcher TortoiseSVN versucht eine Text-Datei mit Informationen darüber zu laden, ob eine neue Version erhältlich ist. Sie können diesen Schlüssel auch unter `HKLM` statt `HKCU` erstellen, aber `HKCU` überschreibt die Einstellung in `HKLM`. Dies ist nützlich in Fällen, in denen der Administrator nicht möchte dass Benutzer TortoiseSVN selber aktualisieren, bevor der Administrator die Version freigegeben hat.

Dateinamen ohne Erweiterung in automatisch Vervollständigen Liste aufnehmen

Die automatisch Vervollständigen Liste im Übertragen-Dialog enthält die Namen der zu übertragenden Dateien. Um die Dateinamen zusätzlich ohne Erweiterung in die Liste aufzunehmen, erzeugen Sie einen `DWORD` Schlüssel namens `HKCU\Software\TortoiseSVN\AutocompleteRemovesExtensions` mit dem Wert 1.

Explorer Spalten überall verfügbar

Die Extraspalten, die TortoiseSVN zur Detailansicht des Explorers hinzufügt, stehen normalerweise nur in Arbeitskopien zur Verfügung. Wenn Sie überall auf diese Spalten zugreifen möchten, erzeugen Sie einen `DWORD` Schlüssel namens `HKCU\Software\TortoiseSVN\ColumnsEveryWhere` mit dem Wert 1.

Trennzeichen für Integrationslogs

Wenn Sie Revisionen von einem anderen Zweig zusammenführen und die Funktionen zum Protokollieren der Datenintegration zur Verfügung stehen, werden die Logmeldungen der zusammengeführten Revisionen zu einer neuen Logmeldung zusammengefasst. Eine vordefinierte Zeichenkette wird zum Trennen der einzelnen Logmeldungen verwendet. Wenn Sie möchten, können Sie einen individuellen Trenner angeben, indem Sie einen SZ Schlüssel namens `HKCU\Software\TortoiseSVN\MergeLogSeparator` anlegen, der die Zeichenkette Ihrer Wahl enthält.

Stets mit TortoiseMerge annotieren

TortoiseSVN ermöglicht es Ihnen, externe Vergleichsprogramme zu verwenden. Die meisten dieser Programme sind allerdings nicht dazu in der Lage, Änderungen zu annotieren ([Abschnitt 4.23.2](#), „**Unterschiede annotieren**“), so dass Sie dafür wieder TortoiseMerge verwenden möchten. Dazu erzeugen Sie einen DWORD Schlüssel namens `HKCU\Software\TortoiseSVN\DiffBlamesWithTortoiseMerge` mit dem Wert 1.

Aktuelle Revision von Ordnern im Log-Dialog hervorheben

Der Log-Dialog hebt die aktuelle Revision einer Datei in der Arbeitskopie durch Fettdruck hervor. Um diese Anzeige auch für einen Ordner zu erhalten, muss die Arbeitskopie durchsucht werden, was Standard ist, aber bei großen Arbeitskopien viel Zeit in Anspruch nimmt. Wenn Sie diese Funktion ändern wollen, müssen Sie einen DWORD unter `HKCU\Software\TortoiseSVN\RecursiveLogRev` in der Registrierung anlegen. Der Wert 0 deaktiviert die Funktion (keine Hervorhebung von Ordnern), der Wert 1 (Standard) holt den Status rekursiv (findet die höchste Revision in der Arbeitskopie) und der Wert 2 prüft die Revision des Ordners selbst, aber keine Unterobjekte.

Erzwingt ein Fehlschlagen eines Checkouts wenn eine Datei/Ordner mit demselben Namen bereits existiert

Wenn Sie eine Arbeitskopie über eine bestehende nicht versionierte Ordnerstruktur auschecken (z.B. nach einem Import), dann werden Dateien, welche nicht mit dem Projektarchiv identisch sind, unverändert belassen aber als geändert markiert. Beim nächsten Übertragen wird die lokale Arbeitskopie dann ins Projektarchiv übertragen. Einige Leute hätten es lieber, dass ein Checkout fehlschlägt, wenn Unterschiede zum Projektarchiv festgestellt werden, so dass, wenn zwei Personen dieselbe Datei hinzufügen, die Übertragung der ersten Person die Datei der zweiten Person nicht überschreibt. Wenn Sie dieses Verhalten bevorzugen, können Sie einen Registrierungsschlüssel unter `HKCU\Software\TortoiseSVN\AllowUnversionedObstruction` als DWORD Wert erstellen und diesen auf 0 setzen.

4.30.11. Subversion Arbeitsordner

VS.NET kann, wenn es mit Webprojekten genutzt wird, nicht mit den `.svn` Ordnern umgehen, die Subversion verwendet, um seine internen Informationen abzulegen. Das ist kein Fehler in Subversion. Der Fehler liegt in VS.NET 2003 und den von ihm verwendeten Frontpage Erweiterungen.

Beachten Sie bitte, dass dieser Fehler in VS2005 und neueren Versionen behoben ist.

Seit Version 1.3.0 von Subversion und TortoiseSVN, können Sie die Umgebungsvariable `SVN_ASP_DOT_NET_HACK` setzen. Wenn diese Variable gesetzt ist, wird Subversion `_svn` Ordner anstelle von `.svn` für seine Administrationsordner verwenden. Sie müssen den Explorer neu starten, damit diese Änderung wirksam wird. Normalerweise erfordert das einen Neustart Ihres PC. Um das zu vereinfachen, können Sie in den allgemeinen Einstellungen eine Option setzen. Siehe [Abschnitt 4.30.1](#), „**Allgemeine Einstellungen**“.

Für weitere Informationen und Möglichkeiten, das Problem von vornherein zu vermeiden, lesen Sie bitte den entsprechenden Artikel in unserer [FAQ](http://tortoisesvn.net/node/aspdotnethack) [http://tortoisesvn.net/node/aspdotnethack].

4.31. Letzter Schritt

Spenden!

Obwohl TortoiseSVN und TortoiseMerge für Sie als Anwender kostenlos sind, können Sie uns unterstützen. Sie können uns Patches schicken und damit bei der Entwicklung direkt mithelfen, oder Sie können uns bei unserer Arbeit motivieren.

Während wir an TortoiseMerge und TortoiseSVN arbeiten hören wir gerne Musik. Und weil wir viele viele Stunden an TortoiseSVN arbeiten, benötigen wir immer wieder *neue* Musik. Aus diesem Grund haben wir im Internet Seiten eingerichtet, von denen aus Sie uns Musik-CD's und auch mal eine DVD schenken können: <http://tortoisesvn.tigris.org/donate.html> [<http://tortoisesvn.tigris.org/donate.html>]. Bitte werfen Sie auch einen Blick auf die Liste der Personen, die das Projekt z.B. durch Übersetzungen unterstützt haben.

Kapitel 5. Das SubWCRev Programm

SubWCRev ist ein Windows Kommandozeilenprogramm, das dazu verwendet werden kann, den Status einer Subversion Arbeitskopie zu ermitteln und optional Schlüsselwörter in einer Vorlagendatei zu ersetzen. Dies wird häufig als Teil eines automatisierten Erstellungsprozesses verwendet, um Informationen über die Arbeitskopie in das erstellte Objekt einfließen zu lassen. Beispielsweise kann man damit die Revisionsnummer in einen „Über...“-Dialog einbetten.

5.1. Die SubWCRev Kommandozeile

SubWCRev liest den Subversion Status aller Dateien einer Arbeitskopie. Externals werden standardmäßig ausgeschlossen. Es ermittelt die höchste Revisionsnummer sowie deren Zeitstempel. Außerdem wird festgehalten, ob es lokale Änderungen in der Arbeitskopie sowie gemischte Revisionsnummern aus verschiedenen Aktualisierungen gibt. Die Revisionsnummer, der Revisionsbereich und der Änderungsstatus werden auf die Standardausgabe ausgegeben.

SubWCRev.exe wird per Kommandozeile oder Skript aufgerufen und besitzt folgende Parameter:

```
SubWCRev ArbeitsKopiePfad [QuelleDatei ZielDatei] [-nmdfe]
```

ArbeitsKopiePfad ist der Pfad der zu prüfenden Arbeitskopie. Sie können SubWCRev nur auf Arbeitskopien und nicht direkt auf Projektarchive anwenden. Der Pfad kann absolut oder relativ zum aktuellen Arbeitsverzeichnis sein.

Wenn Sie möchten, dass SubWCRev Schlüsselwortersetzungen durchführt, so dass Felder wie Revisionsnummer und URL in einer Textdatei gespeichert werden, müssen Sie eine Schablone QuelleDatei und eine Ausgabedatei ZielDatei angeben, die die ersetzte Version der Schablone enthält.

Es gibt mehrere Kommandozeilenooptionen, die die Arbeitsweise von SubWCRev beeinflussen. Wenn Sie mehr als einen verwenden, müssen die Optionen in einer einzelnen Gruppe angegeben werden, z.B. -nm, nicht -n -m.

Wechseln zu	Beschreibung
-n	Wenn dieser Schalter angegeben wurde, gibt SubWCRev ERRORLEVEL 7 zurück, falls die Arbeitskopie lokale Modifikationen enthält. Damit kann das Erzeugen mit nicht übertragenen Änderungen vermieden werden.
-m	Wenn dieser Schalter angegeben wurde, gibt SubWCRev ERRORLEVEL 8 zurück, falls die Arbeitskopie gemischte Revisionen enthält. Damit kann das Erzeugen mit einer nur teilweise aktuellen Arbeitskopie vermieden werden.
-d	Wenn dieser Schalter angegeben wurde, gibt SubWCRev ERRORLEVEL 9 zurück, falls die Zieldatei bereits existiert.
-f	Wenn dieser Schalter angegeben wurde, wird SubWCRev die „last-changed“ Revision von Ordnern beachten. Die Vorgabe ist, dass nur Dateien zum Ermitteln der Revisionsnummern herangezogen werden.
-e	Wenn dieser Schalter angegeben wurde, wird SubWCRev Verzeichnisse überprüfen, die mittels svn:externals eingebunden wurden, falls diese auf dasselbe Projektarchiv verweisen. Die Vorgabe ist, externe Verweise zu ignorieren.
-x	Wenn dieser Schalter angegeben wurde, gibt SubWCRev die Revisionsnummern in HEX aus.
-X	Wenn dieser Schalter angegeben wurde, gibt SubWCRev die Revisionsnummern in HEX mit vorangestelltem '0X' aus.

Tabelle 5.1. Liste der Kommandozeilenooptionen

5.2. Schlüsselwörterersetzung

Wenn eine Quell- und eine Zieldatei angegeben werden, kopiert SubWCRev die Quelldatei zur Zieldatei und ersetzt dabei die folgenden Schlüsselwörter:

Schlüsselwort	Beschreibung
\$WCREV\$	Wird durch die höchste Revisionsnummer in der Arbeitskopie ersetzt.
\$WCDATE\$	Wird durch Datum/Zeit der höchsten übertragenen Revision ersetzt. Als Vorgabe wird das internationale Format genutzt: <code>jjjj-mm-tt hh:mm:ss</code> . Alternativ können Sie ein eigenes Format angeben, das mittels <code>strftime()</code> formatiert wird, z.B. <code>\$WCDATE=%a %b %d %I:%M:%S %p\$</code> . Eine Liste der verfügbaren Formatieranweisungen finden Sie unter folgender Onlinereferenz [http://www.cppreference.com/stddate/strftime.html].
\$WCNOW\$	Wird durch Datum/Zeit der Systemuhr ersetzt. Kann zum Beispiel dazu dienen, den Zeitpunkt der Erstellung festzuhalten. Die Formatierung erfolgt wie bei \$WCDATE\$ beschrieben.
\$WCRANGES\$	Wird durch den Revisionsnummernbereich der Arbeitskopie ersetzt. Wenn sich die Arbeitskopie in einem konsistenten Zustand befindet, eine einzelne Revisionsnummer. Wenn die Arbeitskopie gemischte Revisionen enthält, weil sie entweder nicht aktuell ist oder weil gezielt auf eine bestimmte Revision aktualisiert wurde, wird ein Bereich in der Form <code>100:200</code> ausgegeben.
\$WCMIXED\$	<code>\$WCMIXED?TText:FText\$</code> wird durch <code>TText</code> ersetzt, falls die Arbeitskopie gemischte Revisionen enthält oder <code>FText</code> , falls nicht.
\$WCMODS\$	<code>\$WCMODS?TText:FText\$</code> wird durch <code>TText</code> ersetzt, falls die Arbeitskopie lokale Änderungen enthält oder <code>FText</code> , falls nicht.
\$WCURL\$	Wird durch die URL des Projektarchivs ersetzt, auf das die an SubWCRev übergebene Arbeitskopie zeigt.
\$WCINSVN\$	<code>\$WCINSVN?TText:FText\$</code> wird durch <code>TText</code> ersetzt, falls der Eintrag versioniert ist oder <code>FText</code> , falls nicht.
\$WCNEEDSLOCK\$	<code>\$WCNEEDSLOCK?TText:FText\$</code> wird durch <code>TText</code> ersetzt, falls der Eintrag die <code>svn:needs-lock</code> Eigenschaft hat oder <code>FText</code> , falls nicht.
\$WCISLOCKED\$	<code>\$WCISLOCKED?TText:FText\$</code> wird durch <code>TText</code> ersetzt, falls der Eintrag gesperrt ist oder <code>FText</code> , falls nicht.
\$WCLOCKDATES\$	Wird durch das Sperrdatum ersetzt. Die Formatierung erfolgt wie bei \$WCDATE\$ beschrieben.
\$WCLOCKOWNER\$	Wird den Namen des Sperreigners ersetzt.
\$WCLOCKCOMMENT\$	Wird durch den Sperrkommentar ersetzt.

Tabelle 5.2. Liste der Kommandozeilenoptionen



Tipp

Einige dieser Schlüsselwörter beziehen sich auf einzelne Dateien, statt auf ganze Arbeitskopien. Deshalb ergeben diese auch nur einen Sinn, wenn SubWCRev für eine einzelne Datei aufgerufen wird. Dies ist der Fall für `literal>$WCINSVN$`

5.3. Beispiele für Schlüsselwörter

Das Beispiel zeigt, wie Schlüsselwörter aus einer Vorlagendatei in der Ausgabedatei ersetzt werden.

```
// Testdatei für SubWCRev: testfile.tmpl

char *Revision = "$WCREV$";
char *Modified = "$WCMODS?Geändert:Nicht geändert$";
char *Date      = "$WCDATE$";
char *RevRange = "$WCRANGE$";
char *Mixed     = "$WCMIXED?Gemischte Revisionen:Nicht gemischt$";
char *URL       = "$WCURL$";

#if $WCMODS?1:0$
#error Quelle ist geändert
#endif

// Ende der Datei
```

Nachdem Sie SubWCRev.exe path\to\workingcopy testfile.tmpl testfile.txt aufgerufen haben, sieht die Ausgabedatei testfile.txt folgendermaßen aus:

```
// Testdatei für 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
```



Tipp

Eine solche Datei ist meist in den Erstellungsprozess integriert und man würde normalerweise annehmen, dass die Datei dann auch versioniert sein sollte. Stellen Sie sicher, dass Sie nur die Vorlage und nicht die generierte Datei versionieren. Andernfalls müssen Sie nach jedem Generieren die Änderungen an der Versionsdatei übertragen, was in der Folge bedeutet, dass Sie die Versionsdatei neu generieren müssen.

5.4. COM Schnittstelle

Wenn sie Zugriff auf Subversion Revisionsinformationen aus anderen Programmen benötigen, können Sie die COM Schnittstelle von SubWCRev benutzen. Das zu erzeugende Objekt heißt SubWCRev.object und unterstützt die folgenden Methoden:

Method	Beschreibung
.GetWCInfo	Diese Methode durchläuft die Arbeitskopie und ermittelt die Revisionsinformationen. Sie muss vor den Methoden aufgerufen werden, die auf diese Informationen zugreifen sollen. Der erste Parameter ist der Pfad. Der zweite Parameter sollte true sein, falls Sie Ordnerrevisionen einschließen wollen. Entspricht dem Schalter -

Method	Beschreibung
	f. Der dritte Parameter sollte true sein, falls Sie svn:externals einschließen wollen. Entspricht dem Schalter -e.
.Revision	Die höchste übertragene Revision in der Arbeitskopie. Entspricht \$WCREV\$.
.Date	Datum und Uhrzeit der höchsten übertragenen Revision. Entspricht \$WCDATE\$
.Author	Der Autor der höchsten übertragenen Revision, das heißt der Person, die die Änderungen in die Arbeitskopie übertragen hat.
.MinRev	Die minimale aktualisierte Revision, wie in \$WCRANGE\$ angezeigt.
.MaxRev	Die maximale aktualisierte Revision, wie in \$WCRANGE\$ angezeigt.
.HasModifications	True, falls es lokale Änderungen gibt.
.Url	Wird durch die URL des Projektarchivs ersetzt, auf das die an GetWCInfo übergebene Arbeitskopie zeigt. Entspricht \$WCURL\$.
.IsSvnItem	True falls das Objekt versioniert ist.
.NeedsLocking	True falls die svn:needs-lock Eigenschaft des Objekts gesetzt ist.
.IsLocked	True falls das Objekt gesperrt ist.
.LockCreationDate	Zeichenkette, die das Datum enthält an dem die Sperre erzeugt wurde. Leer, falls das Objekt nicht gesperrt ist.
.LockOwner	Zeichenkette, die den Eigentümer der Sperre enthält. Leer, falls das Objekt nicht gesperrt ist.
.LockComment	Die Meldung, die eingegeben wurde als die Sperre erzeugt wurde.

Tabelle 5.3. Unterstützte COM-Automatisierungen

Das folgende Beispiel zeigt, wie die Schnittstelle genutzt werden kann.

```
// testCOM.js - javascript file
// Testskript für das SubWCRev COM/Automatisierungsobjekt

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);
```

Kapitel 6. IBugtraqProvider Schnittstelle

Um Fehlerverfolgungssysteme besser als durch die `bugtraq`: Eigenschaften ansteuern zu können, verfügt TortoiseSVN über eine COM Schnittstelle. Mit COM Modulen ist es möglich, Informationen direkt aus dem Fehlerverfolgungssystem abzurufen, mit dem Anwender zu kommunizieren und Informationen, z.B. über offene Punkte an TortoiseSVN zu liefern. Weiterhin können Logmeldungen überprüft und es können sogar nach einer erfolgreichen Übertragung Aktionen, z.B. das Schließen eines offenen Punktes, durchgeführt werden.

Wir können Ihnen keine Anleitung liefern, wie Sie ein COM Objekt in Ihrer bevorzugten Programmiersprache erstellen, aber wir haben einige Beispielmole in C++/ATL und C# in dem Ordner `contrib/issue-tracker-plugins` unseres Projektarchivs. In diesem Ordner finden sich auch die nötigen Includedateien, die Sie zum Erstellen Ihres Moduls benötigen. ([Abschnitt 3](#), „*TortoiseSVN ist frei!*“ erklärt, wie Sie auf das Projektarchiv zugreifen.

6.1. Die IBugtraqProvider Schnittstelle

TortoiseSVN 1.5 unterstützt Module, die die IBugtraqProvider Schnittstelle implementieren. Diese Schnittstelle stellt die folgenden Methoden für die Kommunikation mit dem Fehlerverfolgungssystem zur Verfügung:

```
HRESULT ValidateParameters (  
    // Elternfenster für die Bedienoberfläche des Moduls,  
    // die während der Überprüfung angezeigt wird.  
    [in] HWND hParentWnd,  
  
    // Die zu überprüfenden Parameter als Zeichenkette.  
    [in] BSTR parameters,  
  
    // Sind die Parameter gültig?  
    [out, retval] VARIANT_BOOL *valid  
);
```

Diese Methode wird vom Einstellungsdialog aufgerufen in dem der Anwender ein Modul hinzufügen und konfigurieren kann. Die Zeichenkette `parameters` kann von dem Modul verwendet werden, um zusätzliche Informationen, z.B. die URL des Fehlerverfolgungssystems, Anmeldeinformationen, etc. zu erhalten. Das Modul sollte die `parameters` Zeichenkette überprüfen und einen Fehlerdialog anzeigen, falls die Zeichenkette ungültig ist. Der `hParentWnd` Parameter sollte für jeden Dialog verwendet werden, der vom Modul als Elternfenster angezeigt wird. Das Modul muss `TRUE` zurückgeben, falls `parameters` gültig ist. Wenn das Modul `FALSE` zurückgibt, lässt der Einstellungsdialog den Anwender das Modul nicht zum Pfad der Arbeitskopie hinzufügen.

```
HRESULT GetLinkText (  
    // Elternfenster für die Bedienoberfläche des Moduls.  
    [in] HWND hParentWnd,  
  
    // Die Parameter als Zeichenkette, für den Fall, dass Sie mit  
    // Ihrem Web-Dienst Rücksprache halten müssen.  
    [in] BSTR parameters,  
  
    // Welchen Text wollen Sie anzeigen?  
    // Verwenden Sie die Locale des aktuellen Threads.
```

```
[out, retval] BSTR *linkText
);
```

Das Modul kann mittels diese Methode einen Text zur Verfügung stellen, der im TortoiseSVN Übertragen-Dialog auf der Schaltfläche, die das Modul aufruft, angezeigt wird, z.B. 'Eintrag wählen'. Stellen Sie sicher, dass die Zeichenkette nicht zu lang ist, da sie sonst unter Umständen nicht auf die Schaltfläche passt. Wenn die Methode einen Fehler, z.B. E_NOTIMPL zurückgibt, wird ein Standardtext auf der Schaltfläche angezeigt.

```
HRESULT GetCommitMessage (
    // Elternfenster für die Bedienoberfläche des Moduls.
    [in] HWND hParentWnd,

    // Parameter für Ihr Modul
    [in] BSTR parameters,
    [in] BSTR commonRoot,
    [in] SAFEARRAY(BSTR) pathList,

    // Der Text, der bereits in der Logmeldung steht.
    // Ihr Modul sollte diesen Text, wo angebracht,
    // in die neue Meldung einfügen.
    [in] BSTR originalMessage,

    // Der neue Text für die Logmeldung.
    // Ersetzt den Originaltext.
    [out, retval] BSTR *newMessage
);
```

Dies ist die Hauptmethode des Moduls. Sie wird aus dem TortoiseSVN Übertragen-Dialog heraus aufgerufen, wenn der Anwender auf die Modul-Schaltfläche klickt. Die Zeichenkette `parameters` muss vom Anwender in den Einstellungen des Moduls angegeben werden. Normalerweise würde ein Modul damit die URL des Fehlerverfolgungssystems sowie die Anmeldeinformationen ermitteln. Die Variable `commonRoot` enthält den Elternpfad aller Objekte, die beim Aufruf des des Übertragen-Dialogs markiert waren. Beachten Sie bitte das dies *nicht* der Basispfad der Objekte ist, die der Anwender im Übertragen-Dialog selbst markiert hat. Der Parameter `pathList` enthält ein Feld von Pfaden (als Zeichenketten) die der Anwender für das Übertragen gewählt hat. `originalMessage` enthält die Logmeldung die der Anwender im Übertragen-Dialog eingegeben hat. Wenn der Anwender nichts eingegeben hat ist diese Zeichenkette leer. Der Rückgabewert `newMessage` überschreibt den Inhalt der Logmeldung im Übertragen-Dialog. Falls das Modul die `originalMessage` nicht verändert, muss es dieselbe Zeichenfolge in `newMessage` zurückliefern, da sonst der vom Anwender eingegebene Text verloren geht.

6.2. Die IBugtraqProvider2 Schnittstelle

In TortoiseSVN 1.6 wurde eine neue Schnittstelle eingeführt, die mehr Funktionalität für die COM Module zur Verfügung stellt. Die IBugtraqProvider2 Schnittstelle erbt von IBugtraqProvider.

```
HRESULT GetCommitMessage2 (
    // Elternfenster für die Bedienoberfläche des Moduls.
    [in] HWND hParentWnd,

    // Parameter für Ihr Modul
    [in] BSTR parameters,
    // Die gemeinsame URL für die Übertragung
    [in] BSTR commonURL,
    [in] BSTR commonRoot,
    [in] SAFEARRAY(BSTR) pathList,
```

```
// Der Text, der bereits in der Logmeldung steht.
// Ihr Modul sollte diesen Text, wo angebracht,
// in die neue Meldung einfügen.
[in] BSTR originalMessage,

// Sie können einer Übertragung eigene Reversionseigenschaften
// zuweisen, indem Sie die nächsten beiden Parameter setzen.
// ACHTUNG: Beide safearrays müssen dieselbe Länge haben.
//           Für jeden Eigenschaftsnamen muss es einen Wert geben!

// Der Inhalt des bugID Feldes (falls angezeigt)
[in] BSTR bugID,

// Der geänderte Inhalt des bugID Feldes
[out] BSTR * bugIDOut,

// Die Liste der Eigenschaftsnamen.
[out] SAFEARRAY(BSTR) * revPropNames,

// Die Liste der Eigenschaftswerte.
[out] SAFEARRAY(BSTR) * revPropValues,

// Der neue Text für die Logmeldung.
// Ersetzt den Originaltext.
[out, retval] BSTR * newMessage
);
```

Diese Methode wird von TortoiseSVN aus dem Übertragen-Dialog aufgerufen, wenn der Benutzer auf die Modulschaltfläche klickt. Sie wird anstelle von `GetCommitMessage()` aufgerufen. Ein Teil der Parameter wurde bereits bei `GetCommitMessage` beschrieben. Bitte schauen Sie dort nach. Der Parameter `commonURL` ist die Eltern-URL aller Pfade, welche beim Aufruf des Übertragen-Dialogs ausgewählt waren. Dies ist im Grunde genommen die URL des `commonRoot` Pfades. Der Parameter `bugID` enthält den Inhalt des Bug-ID Eingabefeldes (sofern diese, konfiguriert mittels `bugtraq:message`, angezeigt wird). Der Rückgabewert `bugIDOut` wird benutzt um das Bug-ID Eingabefeld zu füllen, wenn die Methode beendet wird. Die Rückgabewerte `revPropNames` und `revPropValues` können Paare von Namen/Werten für Revisions-Eigenschaften enthalten, welche zusammen mit der Übertragung gesetzt werden sollen. Ein Modul muss sicherstellen, dass beide Felder die gleiche Grösse haben! Jeder Eigenschaftsname in `revPropNames` muss einen entsprechenden Wert in `revPropValues` enthalten. Falls keine Reversionseigenschaften gesetzt werden sollen, muss das Modul leere Felder zurückliefern.

```
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
);
```

Diese Methode wird aufgerufen kurz bevor der Übertragen-Dialog geschlossen wird und die eigentliche Übertragung beginnt. Ein Modul kann mit Hilfe dieser Methode die Logmeldung und/oder die zu übertragenden Dateien/Ordner überprüfen und die Übertragung entweder zulassen oder blockieren. Die Parameter sind dieselben wie für `GetCommitMessage2()`, mit dem Unterschied dass `commonURL` die gemeinsame URL aller *ausgewählten* Dateien/Ordner und `commonRoot` der gemeinsame Pfad aller ausgewählten Dateien/Ordner ist. Der Rückgabewert `errorMessage` muss entweder eine

Fehlermeldung enthalten, welche TortoiseSVN dem Benutzer anzeigt oder leer sein, um die Übertragung zuzulassen. Wenn eine Fehlermeldung zurückgegeben wird, zeigt TortoiseSVN diese dem Benutzer in einem Fehlerdialog an und hält den Übertragen-Dialog offen, so dass der Benutzer den Fehler korrigieren kann. Ein Modul sollte deshalb in der Fehlermeldung mitteilen, *was* genau falsch ist und wie der Benutzer dies korrigieren kann.

```
HRESULT OnCommitFinished (
    // Elternfenster für die Bedienoberfläche des Moduls.
    [in] HWND hParentWnd,

    // Die gemeinsame Basis aller übertragenen Pfade.
    [in] BSTR commonRoot,

    // Alle übertragenen Pfade.
    [in] SAFEARRAY(BSTR) pathList,

    // Der Text, der bereits in der Logmeldung steht.
    [in] BSTR logMessage,

    // Die Revision der Übertragung.
    [in] ULONG revision,

    // Ein Fehler, der dem Anwender angezeigt wird, falls
    // die Funktion etwas anderes als S_OK zurück gibt.
    [out, retval] BSTR * error
);
```

Diese Methode wird nach einer erfolgreichen Übertragung aufgerufen. Ein Modul kann diese Methode verwenden, um zum Beispiel einen offenen Eintrag zu schließen oder weitere Informationen an den Eintrag anzuhängen. Die Parameter sind die gleichen wie bei `GetCommitMessage2`.

```
HRESULT HasOptions(
    // Falls das Modul einen eigenen Einstellungsdialog hat.
    [out, retval] VARIANT_BOOL *ret
);
```

Diese Methode wird aus dem Einstellungsdialog für Module heraus aufgerufen. Falls ein Modul einen eigenen Einstellungsdialog über `ShowOptionsDialog` bereitstellt, muss es `TRUE` zurückliefern, sonst `FALSE`.

```
HRESULT ShowOptionsDialog(
    // Elternfenster für den Einstellungsdialog des Moduls.
    [in] HWND hParentWnd,

    // Parameter für Ihr Modul.
    [in] BSTR parameters,

    // Die neuen Parameter als Zeichenkette
    [out, retval] BSTR * newparameters
);
```

Diese Methode wird im Einstellungsdialog aufgerufen, wenn der Anwender auf die 'Optionen' Schaltfläche klickt, die angezeigt wird, wenn `HasOptions` `TRUE` zurückgibt. Ein Modul kann einen Einstellungsdialog zur Verfügung stellen, um dem Anwender die Einrichtung zu vereinfachen.

Die Zeichenkette `parameters` enthält die bisher eingetragenen Parameter des Moduls. Der Rückgabewert `newparameters` muss die neuen Parameter enthalten, die das Modul aus dem Einstellungsdialog ermittelt hat. Die Zeichenkette `parameters` wird an alle anderen `IBugtraqProvider` und `IBugtraqProvider2` Methoden übergeben.

Anhang A. Häufig gestellte Fragen (FAQ)

Da TortoiseSVN sich ständig weiterentwickelt, ist es manchmal schwierig, die Dokumentation auf dem neuesten Stand zu halten. Wir unterhalten eine [Online FAQ](http://tortoisesvn.tigris.org/faq.html) [http://tortoisesvn.tigris.org/faq.html], die eine Auswahl der am häufigsten auf den TortoiseSVN Mailinglisten <dev@tortoisesvn.tigris.org> und <users@tortoisesvn.tigris.org> gestellten Fragen enthält.

Wir unterhalten außerdem ein [Fehlerverfolgungssystem](http://issues.tortoisesvn.net) [http://issues.tortoisesvn.net], mit dessen Hilfe Sie sich einen Überblick über die Liste der offenen Punkte sowie über bereits beseitigte Fehler verschaffen können. Wenn Sie glauben, einen Fehler gefunden zu haben oder wenn Sie einen Verbesserungsvorschlag haben, schauen Sie bitte erst dort nach, ob das Problem nicht bereits dokumentiert ist.

Wenn Sie eine Frage haben, auf die Sie nirgends eine Antwort gefunden haben, sollten Sie die Frage in einer unserer Mailingliste stellen. <users@tortoisesvn.tigris.org> ist für Fragen zur Arbeit mit TortoiseSVN. Wenn Sie bei der Entwicklung helfen wollen, sollten Sie an den Diskussionen auf <dev@tortoisesvn.tigris.org> teilnehmen.

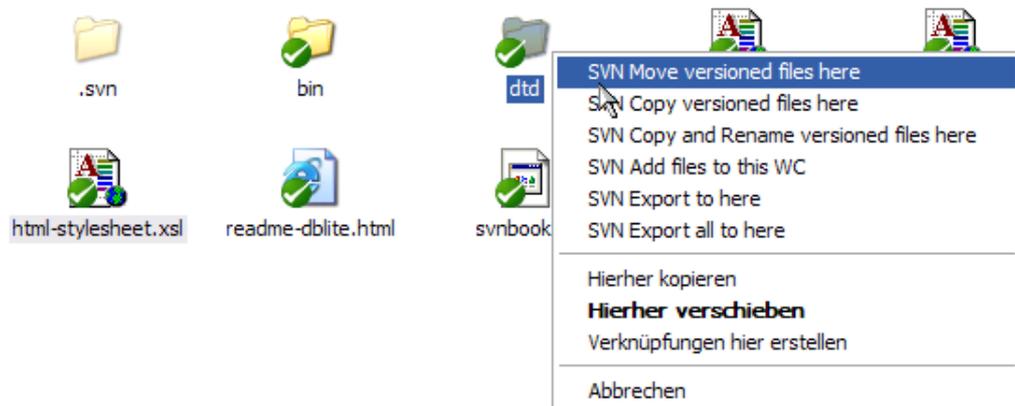
Anhang B. Wie kann ich...

Dieser Anhang enthält Lösungen zu einigen Problemen / Fragen, die bei der Arbeit mit TortoiseSVN auftreten können.

B.1. Viele Dateien auf einmal verschieben / kopieren

Einzelne Dateien können auch mit Hilfe des TortoiseSVN → Umbenennen... Befehls verschoben werden, aber wenn Sie viele Dateien verschieben wollen, ist das zu langsam und zu viel Arbeit.

Der empfohlene Weg ist, die Dateien mittels Rechts-Ziehen an den gewünschten Ort zu befördern. Machen Sie einen Rechtsklick auf die Dateien, die Sie verschieben / kopieren wollen, ohne die rechte Maustaste loszulassen. Dann ziehen Sie die Dateien an den neuen Ort und lassen die rechte Maustaste los. Ein Kontextmenü erscheint, aus dem Sie entweder Kontextmenü → SVN Dateien hierher kopieren oder Kontextmenü → SVN Dateien hierher verschieben wählen können.



B.2. Anwender zwingen eine Logmeldung einzugeben

Es gibt zwei Wege, die Anwender dazu zu bringen, eine Logmeldung einzugeben. Eine Methode geht nur mit TortoiseSVN, die andere funktioniert mit allen Subversion Clients, erfordert jedoch direkten Zugriff auf den Server.

B.2.1. Aktionsskript auf dem Server

Wenn Sie direkten Zugriff auf den Subversion Server haben, können Sie ein `pre-commit` Aktionsskript installieren, das alle Übertragungen mit leeren oder zu kurzen Logmeldungen ablehnt.

Im Projektarchivordner auf dem Server gibt es einen Unterordner namens `hooks`, der die Aktionsskripte enthält. Die Datei `pre-commit.tmpl` enthält ein Beispielskript, das leere Logmeldungen verwirft. Diese Datei enthält auch Hinweise zur Installation des Skriptes. Folgen Sie den Anweisungen in dieser Datei.

Diese Methode wird empfohlen, wenn Ihre Anwender auch andere Subversion Clients als TortoiseSVN benutzen. Der Nachteil ist, dass die Übertragung vom Server abgelehnt wird und dass die Anwender eine Fehlermeldung erhalten. Das Clientprogramm kann vor der Übertragung nicht wissen, dass diese abgelehnt werden wird. Wenn Sie möchten, dass TortoiseSVN die OK Schaltfläche deaktiviert, wenn die Logmeldung nicht lang genug ist, benutzen Sie die unten stehende Methode.

B.2.2. Projekteigenschaft setzen

TortoiseSVN verwendet Eigenschaften, um einige seiner Funktionen zu steuern. Eine davon ist die `tsvn:logminsize` Eigenschaft.

Wenn Sie diese Ordneigenschaft auf einen Zahlenwert setzen, deaktiviert TortoiseSVN so lange die OK Schaltfläche in allen Übertragen-Dialogen, bis der Anwender eine Logmeldung eingegeben hat, die mindestens so lang wie der angegebene Zahlenwert ist.

Für detaillierte Informationen über Projekteigenschaften schlagen Sie bitte in [Abschnitt 4.17, „Projekt-Einstellungen“](#) nach.

B.3. Gezielt Dateien aus dem Projektarchiv aktualisieren

Normalerweise bringen Sie Ihre Arbeitskopie mit TortoiseSVN → Aktualisieren auf den neuesten Stand. Wenn Sie aber nur ein paar Dateien aktualisieren wollen, ohne Änderungen in anderen Dateien zu übernehmen, müssen Sie anders vorgehen.

Wählen Sie TortoiseSVN → Prüfe auf Änderungen und Klicken Sie auf Projektarchiv prüfen, um zu sehen, was sich dort geändert hat. Wählen Sie die Dateien, die Sie aktualisieren wollen und Aktualisieren Sie diese aus dem Kontextmenü des Dialogs.

B.4. Revisionen im Projektarchiv rückgängig machen

B.4.1. Mit Hilfe des Log-Dialogs

Die einfachste Methode, Änderungen einer einzelnen Revision rückgängig zu machen, ist den Log-Dialog zu verwenden. Auf diese Weise können Sie auch neuere Änderungen verwerfen und eine frühere Revision zur HEAD Revision machen.

1. Wählen Sie den Ordner, in dem Sie die Änderungen rückgängig machen wollen. Wenn Sie alle Änderungen rückgängig machen wollen, ist das der oberste Ordner.
2. Wählen Sie TortoiseSVN → Zeige Log, um eine Liste der Revisionen anzuzeigen. Eventuell müssen Sie mit Zeige Alle oder Nächste 100 weitere Revisionen nachladen, um die gewünschte Revision angezeigt zu bekommen.
3. Wählen Sie die Revision, die Sie rückgängig machen wollen. Wenn Sie einen ganzen Bereich verwerfen wollen, halten Sie die **Umsch** Taste gedrückt, während Sie die letzte Revision markieren. Beachten Sie, dass der Revisionsbereich nicht unterbrochen sein darf. Machen Sie einen Rechtsklick auf die gewählten Revision(en) und wählen Sie Kontextmenü → Änderungen dieser Revision rückgängig machen.
4. Wenn Sie eine ältere Revision zur neuen HEAD Revision machen wollen, führen Sie einen Rechtsklick auf die gewählte Revision aus, wählen Kontextmenü → Rückgängig zu dieser Revision. Dies verwirft *alle* Änderungen nach der gewählten Revision.

Sie haben nun die Änderungen in Ihrer Arbeitskopie rückgängig gemacht. Prüfen Sie die Ergebnisse und übertragen Sie die Änderungen.

B.4.2. Mit Hilfe des Zusammenführen-Dialogs

Einen größeren Revisionsbereich machen Sie mit Hilfe des Zusammenführen-Dialogs rückgängig. Die vorherige Methode nutzt das Zusammenführen verdeckt; diese Methode nutzt es explizit, in dem ein Revisionsbereich *rückwärts* mit der Arbeitskopie zusammengeführt wird.

1. Wählen Sie TortoiseSVN → Zusammenführen in Ihrer Arbeitskopie.
2. Im Von: Feld geben Sie die URL des Zweiges an, der die Änderungen enthält, welche Sie in Ihrer Arbeitskopie rückgängig machen wollen. Der Wert sollte bereits als Vorgabe eingetragen sein.
3. In der Von Revision tragen Sie die aktuelle Revisionsnummer ein. Wenn Sie sicher sein können, dass niemand anders in der Zwischenzeit Änderungen überträgt, können Sie die HEAD Revision verwenden.
4. Stellen Sie sicher dass die Nutze „Von:“ URL Option gewählt ist.
5. In die Bis Revision tragen Sie die Revisionsnummer ein bis zu der Sie Änderungen rückgängig machen wollen, das heißt die Nummer vor der ersten rückgängig zu machenden Revision.
6. Klicken Sie auf OK, um das Zusammenführen anzustoßen.

Sie haben nun die Änderungen in Ihrer Arbeitskopie rückgängig gemacht. Prüfen Sie die Ergebnisse und übertragen Sie die Änderungen.

B.4.3. Mit Hilfe von `svndumpfilter`

Da Subversion niemals Daten verliert und alle Änderungen protokolliert sind Ihre „rückgängig gemachten“ Revisionen noch als Zwischenstände im Projektarchiv enthalten. Das kann bei versehentlich in ein öffentlich zugängliches Projektarchiv übertragenen vertraulichen Daten ein großes Problem darstellen. Wenn Sie möchten, dass Ihre Revisionen vollständig aus dem Projektarchiv verschwinden, müssen Sie recht extreme Maßnahmen ergreifen. Solange es keinen wirklich guten Grund dafür gibt, *raten wir dringend davon ab*.

Der einzige Weg, Daten komplett aus dem Projektarchiv zu entfernen führt über das Subversion Kommandozeilen Programm `svnadmin`. Eine Anwendungsbeschreibung findet sich im Kapitel [Repository Maintenance](http://svnbook.red-bean.com/en/1.5/svn.reposadmin.maint.html) [<http://svnbook.red-bean.com/en/1.5/svn.reposadmin.maint.html>].

B.5. Zwei Revisionen einer Datei oder eines Ordners vergleichen

Wenn Sie zwei Revisionen desselben Objekts, zum Beispiel Revisionen 100 und 200 vergleichen wollen, rufen Sie TortoiseSVN → Zeige Log auf, um sich die Historie des Objekts anzeigen zu lassen. Wählen Sie die zwei Revisionen, die Sie vergleichen wollen und wählen Sie Kontextmenü → Revisionen vergleichen.

Wenn Sie Revisionen desselben Objekts in zwei verschiedenen Entwicklungszweigen vergleichen wollen, können Sie im Projektarchivbetrachter beide Zweige öffnen, das Objekt in beiden Zweigen markieren und dann Kontextmenü → Revisionen vergleichen aufrufen.

Wenn Sie zwei vollständige Zweige miteinander vergleichen wollen, verwenden Sie TortoiseSVN → Revisionsgraph. Markieren Sie dort die beiden zu vergleichenden Knoten und wählen Sie Kontextmenü → HEAD Revisionen vergleichen. Dies erzeugt eine Liste der unterschiedlichen Dateien und Sie können sich die Detailunterschiede aus der Liste heraus anschauen. Sie können auch eine Baumstruktur mit allen geänderten Dateien oder einfach eine Liste aller geänderten Dateien exportieren. Mehr dazu finden Sie in [Abschnitt 4.10.3, „Ordner vergleichen“](#). Alternativ können Sie Kontextmenü → Standard-Diff der HEAD Revisionen wählen, um eine Zusammenfassung der Unterschiede mit minimalem Kontext zu erhalten.

B.6. Ein gemeinsames Unterprojekt einbinden

Manchmal ist es notwendig, ein anderes Projekt in Ihre Arbeitskopie einzubinden, zum Beispiel eine externe Bibliothek. Sie möchten aber keine Kopie davon in Ihr Projektarchiv aufnehmen, da Sie dann

die Verbindung zum Original verlieren. Oder Sie haben mehrere Projekte welche Code-Teile gemeinsam haben. Es gibt mindestens drei verschiedene Wege um diese Situation zu meistern.

B.6.1. Die Eigenschaft `svn:externals`

Setzen Sie die `svn:externals` Eigenschaft auf Ordner in Ihrem Projekt. Diese Eigenschaft besteht aus ein oder mehreren Zeilen. Jede Zeile besteht aus dem Namen eines Unterordners den Sie als Ziel verwenden wollen sowie der URL im Projektarchiv, die dorthin ausgecheckt werden soll. Weitere Informationen finden sich in [Abschnitt 4.18, „Externe Objekte“](#).

Übertragen Sie diese Änderungen. Nun, wenn Sie Ihre Arbeitskopie aktualisieren wird Subversion eine Kopie des externen Projektes in Ihre Arbeitskopie einfügen. Jedes Mal wenn Sie eine Aktualisierung vornehmen, wird automatisch auch das externe Projekt aktualisiert. Änderungen, welche Sie am gemeinsamen Code vornehmen werden ebenfalls in das externe Projektarchiv mit übertragen.

Wenn das externe Projekt im selben Projektarchiv ist, werden alle Änderungen in diesem externen Projekt im Übertragen-Dialog aufgelistet und gemeinsam mit dem Hauptprojekt übertragen.

Wenn ein externes Projekt in einem anderen Projektarchiv ist, werden Sie zwar über Änderungen im externen Projekt informiert, Sie müssen diese jedoch separat übertragen.

Von den drei beschriebenen Methoden ist dies die einzige, die keine clientseitige Konfiguration erfordert. Sobald Externals in den Ordneigenschaften festgelegt wurden, werden sämtliche Clients die Ordner beim Aktualisieren erhalten.

B.6.2. Verschachtelte Arbeitskopien

Erstellen Sie einen neuen Ordner in Ihrem Projekt, welcher den gemeinsamen Code enthalten soll, aber fügen Sie diesen nicht zu Ihrer Arbeitskopie hinzu.

Wählen Sie TortoiseSVN → Auschecken auf dem neuen Ordner und checken Sie eine Kopie des gemeinsamen Codes in diesen Ordner aus. Sie haben nun eine separate Arbeitskopie innerhalb Ihrer eigenen Arbeitskopie.

Diese beiden Arbeitskopien sind komplett unabhängig voneinander. Wenn Sie Änderungen übertragen möchten, so werden Änderungen in der zweiten Arbeitskopie nicht mit erfasst. Ebenfalls wird die zweite Arbeitskopie bei einer Aktualisierung nicht mit-aktualisiert. Sie müssen die zweite Arbeitskopie jeweils getrennt aktualisieren oder Änderungen darin übertragen.

B.6.3. Relative Pfade

Wenn Sie denselben Code in mehreren Projekten benutzen und Sie nicht für jedes Projekt das diesen gemeinsamen Code benutzt eine Kopie davon haben möchten, dann können Sie diesen Code auch an eine ganz bestimmte Stelle auschecken, auf welche von allen Projekten welche den Code benutzen zugegriffen werden kann. Zum Beispiel:

```
C:\Projekte\Proj1
C:\Projekte\Proj2
C:\Projekte\Proj3
C:\Projekte\Gemeinsamer_Code
```

dann können Sie in Ihren Projekten mittels relativer Pfade auf diesen Code zugreifen, zum Beispiel `..\..\Gemeinsamer_code\DSPCore`.

Wenn Ihre Projekte jedoch überall verstreut an verschiedenen Orten sind, dann können Sie eine Variante hiervon benutzen: den gemeinsamen Code an einem Ort speichern und diesen Ort dann mittels Laufwerks-Substitution an einen fixen Ort binden. Dann können Sie diesen fixen Pfad in Ihren Projekten direkt einbinden. Zum Beispiel können Sie den gemeinsamen Code an `D:\Dokumente\Framework`

oder C:\Dokumente und Einstellungen\{login}\Eigene Dateien\Framework auschecken und dann mittels

```
SUBST X: "D:\Dokumente\Framework"
```

mit dem Laufwerksbuchstaben X verbinden. In Ihrem Sourcecode können Sie dann auf diese absolute Position zugreifen, zum Beispiel

```
#include "X:\superio.h\superio.h"
```

Diese Methode funktioniert jedoch nur in einer PC-Umgebung, und Sie müssen genau dokumentieren wie die Laufwerks-Verbindung gemacht werden muss damit die Team-Mitglieder die Projekte auch Kompilieren können und die gemeinsamen Dateien auch finden. Diese Methode ist wirklich nur in geschlossene Umgebungen anwendbar und nicht für die normale Verwendung empfohlen.

B.7. Eine Verknüpfung zu einem Projektarchiv erstellen

Wenn Sie den Projektarchivbetrachter immer wieder benötigen, um ein bestimmtes Projektarchiv anzuzeigen, dann können Sie auch eine Verknüpfung auf dem Desktop erstellen und die benötigten Parameter direkt an TortoiseProc.exe übergeben. Erstellen Sie einfach eine Verknüpfung und setzen Sie das Ziel auf:

```
TortoiseProc.exe /command:repobrowser /path:"url/zum/projektarchiv"
```

Sie müssen natürlich eine existierende URL eingeben.

B.8. Dateien ignorieren, die bereits unter Versionskontrolle sind

Wie können Sie Dateien wieder aus der Versionskontrolle entfernen, die Sie fälschlicherweise hinzugefügt haben, ohne die Dateien selbst zu löschen? Vielleicht haben Sie Ihre eigene IDE Konfigurationsdatei, welche zwar nicht Teil des Projekts ist, aber was Sie dennoch einige Zeit gekostet hat, es so zu konfigurieren wie Sie es gerne möchten, hinzugefügt.

Wenn Sie die hinzugefügte Datei noch nicht übertragen haben, dann müssen Sie nur den Befehl TortoiseSVN → Rückgängig... ausführen. Sie sollten dann die Datei(en) zur Liste der ignorierten Dateien hinzufügen damit dies nicht nochmals fälschlicherweise hinzugefügt werden können.

Wenn die Dateien jedoch schon im Projektarchiv gespeichert sind, ist ein wenig mehr Arbeit notwendig.

1. Halten Sie die **Umsch** Taste gedrückt, um das erweiterte Kontextmenü zu erhalten und wählen Sie TortoiseSVN → Löschen (Lokal behalten), um die Datei aus dem Projektarchiv, nicht aber aus der Arbeitskopie zu löschen.
2. TortoiseSVN → Übertragen Sie den Elternordner.
3. Fügen Sie die Datei bzw. den Ordner zur Ignorieren-Liste hinzu, damit Sie nicht nochmals in die gleiche Situation geraten.

B.9. Eine Arbeitskopie aus der Versionskontrolle nehmen

Wenn Sie eine Arbeitskopie in einen normalen Ordner ohne die .svn Steuerverzeichnisse verwandeln wollen, können Sie dies durch einen Export auf sich selbst erreichen. Lesen Sie in [Abschnitt 4.26.1, „Eine Arbeitskopie aus der Versionskontrolle entfernen“](#) nach, wie das geht.

B.10. Eine Arbeitskopie löschen

Wenn Sie eine Arbeitskopie haben, die Sie nicht mehr benötigen, können Sie diese einfach im Windows Explorer löschen. Arbeitskopien sind in sich geschlossene lokale Einheiten, die außerhalb keine Daten ablegen.

Anhang C. Tipps für Administratoren

Dieser Anhang enthält Lösungen zu einigen Problemen / Fragen, die auftreten können, wenn Sie zentral verteilte Installationen von TortoiseSVN vornehmen wollen.

C.1. TortoiseSVN über Gruppenrichtlinien verteilen

Das TortoiseSVN Installationsprogramm ist eine MSI Datei, die Sie normalerweise problemlos in die Gruppenrichtlinien Ihres Domänencontrollers mit aufnehmen können.

Eine gute Anleitung findet sich in Artikel 314934 von Microsofts Wissensbasis: <http://support.microsoft.com/?kbid=314934>.

Versionen 1.3.0 und neuer von TortoiseSVN müssen unter *Computerkonfiguration* und nicht unter *Benutzerkonfiguration* installiert werden. Das liegt daran, dass diese Versionen die neuen CRT und MFC DLLs benötigen, die nur *per computer* und nicht *per Benutzer* eingesetzt werden. Wenn Sie TortoiseSVN wirklich pro Benutzer installieren wollen, müssen Sie zunächst die MFC and CRT Pakete in der Version 8 von Microsoft auf jedem davon betroffenen Rechner installieren.

C.2. Die Versionsprüfung umleiten

TortoiseSVN überprüft alle paar Tage, ob eine neue Version zur Verfügung steht. Wenn das der Fall ist, erscheint ein Dialog, der den Anwender darauf hinweist.

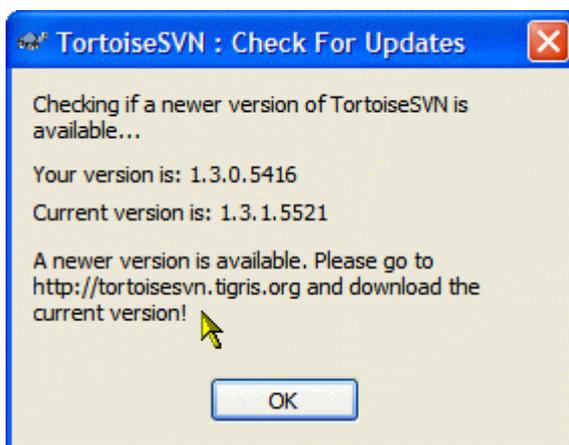


Abbildung C.1. Die Versionsprüfung

Wenn Sie für viele Anwender in Ihrer Domäne zuständig sind, möchten Sie vielleicht, dass die Anwender nur von Ihnen freigegebene Versionen und nicht die jeweils neueste Version installieren. Deshalb sollte der Aktualisieren-Dialog nicht angezeigt werden, damit die Anwender die Software nicht selbst aktualisieren.

Versionen 1.4.0 und neuer von TortoiseSVN ermöglichen es Ihnen die Aktualisierungsprüfung auf einen eigenen Server umzuleiten. Sie können den Registrierungsschlüssel `HKCU\Software\TortoiseSVN\UpdateCheckURL` (Zeichenkette) auf eine URL setzen, die auf eine Textdatei in Ihrem Intranet zeigt. Diese Textdatei muss das folgende Format haben:

```
1.4.1.6000
Eine neue Version von TortoiseSVN steht zur Verfügung!
http://192.168.2.1/downloads/TortoiseSVN-1.4.1.6000-svn-1.4.0.msi
```

Die erste Zeile in der Datei ist das Versionsstring. Sie müssen sicherstellen, dass es exakt mit der Version des TortoiseSVN Installationspakets übereinstimmt. Die zweite Zeile enthält einen freien Text, der im Aktualisieren-Dialog angezeigt wird. Sie können hineinschreiben was sie wollen, sollten jedoch bedenken, dass der Platz im Dialog beschränkt ist. Zu lange Zeilen werden abgeschnitten. Die dritte Zeile enthält die URL des neuen Installationspakets. Diese URL wird geöffnet, wenn der Anwender auf die Meldung im Aktualisieren-Dialog klickt. Sie können den Anwender auch zu einer Webseite anstatt direkt zum MSI Installer leiten. Die URL wird mit dem Standard Web-Browser geöffnet. Wenn Sie eine Webseite angeben, wird diese einfach angezeigt, wenn Sie das msi Paket angeben, wird der Browser den Anwender bitten, die Datei lokal zu speichern.

C.3. Die SVN_ASP_DOT_NET_HACK Umgebungsvariable setzen

Seit Version 1.4.0 und neuer, bietet der TortoiseSVN Installer dem Anwender nicht mehr die Möglichkeit, die SVN_ASP_DOT_NET_HACK Umgebungsvariable zu setzen, da dies viele Probleme verursacht hat und Benutzer verwirrt hat, die grundsätzlich *alles* installieren, auch wenn sie keine Ahnung haben, wozu das gut ist.

Aber diese Option ist nur vor dem Anwender verborgen. Sie können den TortoiseSVN Installer weiterhin dazu zwingen, diese Umgebungsvariable zu setzen, indem Sie die ASPDOTNETHACK Eigenschaft auf TRUE setzen. Zum Beispiel können Sie den Installer wie folgt aufrufen:

```
msiexec /i TortoiseSVN-1.4.0.msi ASPDOTNETHACK=TRUE
```

C.4. Kontextmenüeinträge deaktivieren

Seit Version 1.5.0 können Sie in TortoiseSVN Kontextmenüeinträge deaktivieren (eigentlich nur verstecken). Da diese Funktion nicht leichtfertig, sondern nur mit gutem Grund genutzt werden sollte, steht dafür keine Benutzeroberfläche zur Verfügung und die Einstellungen müssen direkt in der Registrierung vorgenommen werden. Auf diese Weise können bestimmte Befehle vor Anwendern verborgen werden. Bitte beachten Sie, dass nur die Kontextmenüs im *Explorer* verschwinden. Die Befehle stehen weiterhin per Kommandozeile oder auch in anderen Dialogen von TortoiseSVN selbst zur Verfügung!

Die Registrierungsschlüssel, welche die Information über die anzuzeigenden Kontextmenüs enthalten, heißen HKEY_CURRENT_USER\Software\TortoiseSVN\ContextMenuEntriesMaskLow und HKEY_CURRENT_USER\Software\TortoiseSVN\ContextMenuEntriesMaskHigh.

Jeder dieser Registrierungsschlüssel ist ein DWORD Wert, bei dem jedes Bit einem bestimmten Menüeintrag entspricht. Ein gesetztes Bit bedeutet, dass der Menüeintrag deaktiviert ist.

Wert	Menüeintrag
0x0000000000000001	Auschecken
0x0000000000000002	Aktualisieren
0x0000000000000004	Übertragen
0x0000000000000008	Hinzufügen
0x0000000000000010	Rückgängig
0x0000000000000020	Aufräumen
0x0000000000000040	Konflikt lösen
0x0000000000000080	Wechseln zu
0x0000000000000100	Importieren
0x0000000000000200	Exportieren

Wert	Menüeintrag
0x0000000000000400	Projektarchiv hier erstellen
0x0000000000000800	Verzweigen/Markieren
0x0000000000001000	Zusammenführen
0x0000000000002000	Löschen
0x0000000000004000	Umbenennen
0x0000000000008000	Aktualisiere zu Revision
0x0000000000010000	Vergleich
0x0000000000020000	Zeige Log
0x0000000000040000	Konflikt bearbeiten
0x0000000000080000	Umplatzieren
0x0000000000100000	Prüfe auf Änderungen
0x0000000000200000	Ignorieren
0x0000000000400000	Projektarchivbetrachter
0x0000000000800000	Annotieren
0x0000000001000000	Erzeuge Patch
0x0000000002000000	Patch anwenden
0x0000000004000000	Revisionsgraph
0x0000000008000000	Sperren
0x0000000010000000	Sperre freigeben
0x0000000020000000	Eigenschaften
0x0000000040000000	Vergleiche mit URL
0x0000000080000000	Nicht versionierte Dateien löschen
0x2000000000000000	Einstellungen
0x4000000000000000	Hilfe
0x8000000000000000	Über

Tabelle C.1. Menüeinträge und ihre Werte

Beispiel: Um die „Umplatzieren“, „Nicht versionierte Objekte Löschen“ und die „Einstellungen“ Menüeinträge zu deaktivieren, addieren Sie die zu den Einträgen gehörenden Werte zusammen:

```

0x0000000000080000
+ 0x0000000008000000
+ 0x2000000000000000
= 0x2000000008008000

```

Der niedrige DWORD Wert (0x80080000) muss dann in HKEY_CURRENT_USER\Software\TortoiseSVN\ContextMenuEntriesMaskLow, der hohe DWORD Wert (0x20000000) in HKEY_CURRENT_USER\Software\TortoiseSVN\ContextMenuEntriesMaskHigh eingetragen werden.

Um die Kontextmenüeinträge wieder zu aktivieren, löschen Sie einfach die beiden Registrierungsschlüssel.

Anhang D. TortoiseSVN automatisieren

Da alle Befehle von TortoiseSVN über Kommandozeilenparameter beeinflusst werden können, haben Sie die Möglichkeit automatische Prozessen mit Batch Skripten zu erstellen oder Dialoge aus anderen Programmen (z.B. Ihrem Lieblingseditor) heraus aufzurufen.



Wichtig

Beachten Sie, dass TortoiseSVN eine graphische Anwendung ist und dass diese Automatisierungsanleitung Ihnen erklärt, wie man die Dialoge von TortoiseSVN zur Abfrage von Benutzereingaben verwendet. Wenn Sie ein Skript schreiben wollen, das keine Eingaben erfordert, sollten Sie dafür das offizielle Subversion Kommandozeilenprogramm verwenden.

D.1. TortoiseSVN Befehle

Das TortoiseSVN GUI Programm heißt `TortoiseProc.exe`. Alle Befehle werden in der Form `/command:abcd` angegeben, wobei `abcd` der - erforderliche - Befehlsname ist. Die meisten Befehle benötigen zumindest einen Parameter, der in der Form `/path:"some\path"` übergeben wird. In der folgenden Tabelle bezieht sich der Befehl auf den `/command:abcd` Parameter und der Pfad auf den `/path:"some\path"` Parameter.

Da einige Befehle eine Liste von Zielpfaden akzeptieren (z.B. Übertragen mehrerer Dateien), kann der `/path` Parameter mehrere, durch `*` getrennte Pfade enthalten.

TortoiseSVN verwendet Temporärdateien, um Parameter zwischen der Shell-Erweiterung und dem Hauptprogramm zu übergeben. Ab der Version 1.5.0 von TortoiseSVN, entfällt der `/notempfile` Parameter und muss nicht mehr angegeben werden.

Der Fortschrittsdialog, der für Übertragungen, Aktualisierungen und viele weitere Aktionen verwendet wird, bleibt normalerweise geöffnet nachdem die Aktion beendet ist bis der Anwender die OK Schaltfläche betätigt. Dieses Verhalten kann durch Auswahl der entsprechenden Option in den Einstellungen verändert werden. Die Verwendung dieser Einstellung wird den Dialog, unabhängig davon ob der Befehl aus dem TortoiseSVN Kontextmenü oder aus einer Batchdatei heraus aufgerufen wurde, schließen.

Um einen anderen Ort für die Konfigurationsdatei anzugeben, verwenden Sie den `/configdir:"path\to\config\dir"` Parameter. Dies überschreibt alle Vorgaben, auch die aus der Registrierung.

Um den Fortschrittsdialog am Ende einer Aktion automatisch zu schließen, ohne die Standardeinstellung zu berücksichtigen, können Sie den `/closeonend` Parameter übergeben.

- `/closeonend:0` Den Dialog nicht automatisch schließen.
- `/closeonend:1` Schließen, wenn keine Fehler aufgetreten sind.
- `/closeonend:2` Schließen, wenn keine Fehler und keine Konflikte aufgetreten sind.
- `/closeonend:2` Schließen, wenn keine Fehler, Konflikte und Zusammenführungen aufgetreten sind.
- `/closeonend:2` Schließen, wenn keine Fehler, Konflikte und Zusammenführungen bei lokalen Operationen aufgetreten sind.

Die untenstehende Tabelle listet alle Befehle, die mittels TortoiseProc.exe ausgeführt werden können. Wie oben beschrieben, müssen sie in der Form `/command:abcd` aufgerufen werden. In der Tabelle wird der `/command` Präfix weggelassen, um Platz zu sparen.

Befehl	Beschreibung
:about	Öffnet den Über TortoiseSVN-Dialog. Dies ist das Standardverhalten, wenn kein Befehl angegeben wird.
:log	Öffnet den Log-Dialog. Der <code>/path</code> definiert die Datei oder den Ordner, für den das Log angezeigt werden soll. Drei weitere Optionen können angegeben werden: <code>/startrev:xxx</code> , <code>/endrev:xxx</code> und <code>/strict</code>
:checkout	Öffnet den Auschecken-Dialog. <code>/path</code> legt das Zielverzeichnis fest und <code>/url</code> definiert die URL von der ausgecheckt werden soll.
:import	Öffnet den Import-Dialog. Der <code>/path</code> bezeichnet das Verzeichnis mit den zu importierenden Daten.
:update	Aktualisiert die Arbeitskopie in <code>/path</code> zur HEAD Revision. Bei Angabe von <code>/rev</code> wird ein Dialog angezeigt, in dem der Anwender angeben kann, zu welcher Revision aktualisiert werden soll. Um die Anzeige des Dialogs zu vermeiden, geben Sie eine Revisionsnummer an <code>/rev:1234</code> . Weitere Optionen sind <code>/nonrecursive</code> und <code>/ignoreexternals</code> .
:commit	Öffnet den Übertragen-Dialog. Der <code>/path</code> gibt das Zielverzeichnis oder die Liste der zu übertragenden Dateien an. Sie können mit Hilfe des <code>/logmsg</code> Parameters eine Logmeldung übergeben oder, falls Sie das nicht per Kommandozeile tun wollen, <code>/logmsgfile:path</code> verwenden, wobei <code>path</code> auf eine Datei zeigt, die die Logmeldung enthält. Falls Sie ein Fehlerverfolgungssystem nutzen, können Sie das Eingabefeld für die Fehler-ID mittels <code>/bugid:"Hier liegt der Fehler"</code> vorbelegen.
:add	Fügt die Datei(en) in <code>/path</code> der Versionskontrolle hinzu.
:revert	Macht lokale Änderungen in der Arbeitskopie rückgängig. Der <code>/path</code> gibt die Objekte an.
:cleanup	Räumt die Hinterlassenschaften unterbrochener oder fehlgeschlagener Operationen auf und entspermt die Arbeitskopie in <code>/path</code> .
:resolve	Markiert den Konflikt der in <code>/path</code> angegebenen Datei als gelöst. Bei Angabe von <code>/noquestion</code> wird der Konflikt ohne Rückfrage aufgelöst.
:reprocreate	Erstellt ein Projektarchiv in <code>/path</code>
:switch	Öffnet den Wechseln Zu-Dialog. Der <code>/path</code> bezeichnet das Zielverzeichnis.
:export	Exportiert die Arbeitskopie in <code>/path</code> in ein anderes Verzeichnis. Wenn <code>/path</code> auf ein nicht versioniertes Verzeichnis zeigt, wird ein Dialog angezeigt, der eine URL erfragt, die in das in <code>/path</code> angegebene Verzeichnis exportiert wird.
:merge	Öffnet den Zusammenführen-Dialog. Der <code>/path</code> definiert den Zielordner. Zum Zusammenführen eines Revisionsbereichs stehen die folgenden Optionen zur Verfügung: <code>/fromurl:URL</code> , <code>/revrange:string</code> . Zum Zusammenführen zweier Bäume stehen folgende Optionen zur Verfügung: <code>/fromurl:URL</code> , <code>/tourl:URL</code> , <code>/fromrev:xxx</code> und <code>/torev:xxx</code> . Diese tragen Vorgabewerte in die Eingabefelder des Dialogs ein.
:mergeall	Öffnet den Alle Zusammenführen-Dialog. Der <code>/path</code> bezeichnet das Zielverzeichnis.
:copy	Öffnet den Verzweigen/Markieren-Dialog. Der <code>/path</code> gibt die Arbeitskopie an. Der <code>/url</code> ist die Ziel URL. Sie können mit dem <code>/logmsg</code> Parameter eine Logmeldung übergeben oder, falls Sie das nicht per Kommandozeile

Befehl	Beschreibung
	tun wollen, /logmsgfile:path verwenden, wobei path auf eine Datei zeigt, die die Logmeldung enthält.
:settings	Öffnet den Einstellungsdialog
:remove	Entfernt die Datei(en) in /path aus der Versionskontrolle.
:rename	Benennt die Datei in /path um. Der neue Dateiname wird über einen Dialog abgefragt. Um die Frage nach dem Umbenennen ähnlicher Dateien in einem Schritt zu umgehen, übergeben Sie /noquestion.
:diff	Startet das externe, in den TortoiseSVN Einstellungen angegebene Vergleichsprogramm. Der /path spezifiziert die erste Datei. Bei Angabe von /path2, werden diese beiden Dateien verglichen. Ohne Angabe von /path2, wird die Datei in /path mit ihrer BASE Revision verglichen. Um die Revisionsnummern vorzugeben, verwenden Sie /startrev:xxx und /endrev:xxx. Wenn /blame, aber nicht /path2 angegeben ist, wird der Vergleich durchgeführt, indem zunächst die Dateien in den gegebenen Revisionen annotiert werden.
:showcompare	Abhängig von den URLs und den zu vergleichenden Revisionen wird entweder ein Standard-Diff (falls die unified Option gesetzt ist), ein Dialog mit einer Liste der geänderten Dateien oder, falls die URLs auf Dateien zeigen, das Vergleichsprogramm mit diesen Dateien angezeigt. Die Optionen url1, url2, revision1 und revision2 müssen angegeben werden. Die Optionen pegrevision, ignoreancestry, blame und unified sind optional.
:conflicteditor	Startet den in der TortoiseSVN Einstellungen angegebenen Konflikteditor mit der konfliktbehafteten Datei in /path.
:relocate	Zeigt den Umplatzieren-Dialog. Der /path bezeichnet die umzuplatzierende Arbeitskopie.
:help	Öffnet die Hilfedatei
:repostatus	Öffnet den Prüfe auf Änderungen-Dialog. Der /path bezeichnet die Arbeitskopie.
:repobrowser	Startet den Projektarchivbetrachter mit der URL der in /path oder /path angegebenen Arbeitskopie. Mit der Option /rev:xxx kann die vom Projektarchivbetrachter anzuzeigende Revision festgelegt werden. Ohne /rev:xxx, wird die HEAD Revision angezeigt. Falls /path auf eine URL zeigt, wird durch /projectpropertiespath:path/to/wc der Pfad festgelegt, aus dem die Projekteigenschaften gelesen und angewendet werden.
:ignore	Fügt alle Objekte in /path zur Liste der ignorierten Dateien hinzu, das heißt für alle Objekte wird die svn:ignore Eigenschaft gesetzt.
:blame	Öffnet den Annotieren-Dialog mit der in /path angegebenen Datei. Wenn die Optionen /startrev und /endrev angegeben sind, wird der Dialog zur Abfrage des Revisionsbereiches nicht angezeigt, sondern die angegebenen Revisionen verwendet. Wenn die Option /line:nnn angegeben ist, wird TortoiseBlame die Datei in dieser Zeile positionieren. Die Optionen /ignoreeol, /ignorespaces und /ignoreallspaces werden ebenfalls unterstützt.

Befehl	Beschreibung
:cat	Speichert eine Datei von einer /path angegebenen Arbeitskopie oder URL an der in /savepath:path spezifizierten Stelle. Eine Revision kann in /revision:xxx angegeben werden, um eine Datei in einer bestimmten Revision zu holen.
:createpatch	Erzeugt eine Patchdatei für den in /path angegebenen Pfad.
:revisiongraph	Zeigt den Revisionsgraphen für den in /path angegebenen Pfad.
:lock	Sperrt eine oder alle Dateien in einem Verzeichnis. Der Sperren-Dialog wird angezeigt, damit der Anwender einen Kommentar für die Sperre eingeben kann. /path
:unlock	Entsperrt eine oder alle Dateien in einem Verzeichnis. /path
:rebuildiconcache	Rekonstruiert den Windows Symbolspeicher. Verwenden Sie diese Funktion nur, wenn die Windows Symbole beschädigt sind. Ein nicht vermeidbarer Seiteneffekt dieser Aktion ist, dass die Desktopsymbole neu arrangiert werden. /noquestion
:properties	Zeigt den Eigenschaftsdialog für den in /path angegebenen Pfad.

Tabelle D.1. Liste der Befehle und Parameter

Beispiele (die auf einer Zeile eingegeben werden sollten):

```
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 Befehle

Das Programm zum Vergleichen von Bildern stellt ein paar Kommandozeilenbefehle zur Verfügung, mit denen Sie kontrollieren können, in welchen Modus das Programm startet. Die Anwendung selbst heißt TortoiseIDiff.exe.

Die untenstehende Tabelle listet alle Optionen, die per Kommandozeile an TortoiseIDiff übergeben werden können.

Option	Beschreibung
:left	Die im linken Fenster angezeigte Datei
:lefttitle	Ein Titel. Dieser Text wird in der Bildansicht anstelle des vollständigen Dateipfades angezeigt.
:right	Die im rechten Fenster angezeigte Datei
:righttitle	Ein Titel. Dieser Text wird in der Bildansicht anstelle des vollständigen Dateipfades angezeigt.
:overlay	Mit dieser Option werden die Bilder transparent überlagert.
:fit	Mit dieser Option werden die Bilder auf die gleiche Größe skaliert.
:showinfo	Zeigt die Bildinformationen

Tabelle D.2. Liste der Parameter

Beispiele (die auf einer Zeile eingegeben werden sollten):

```
TortoiseIDiff.exe /left:"c:\images\img1.jpg" /lefttitle:"image 1"  
                  /right:"c:\images\img2.jpg" /righttitle:"image 2"  
                  /fit /overlay
```

Anhang E. Befehle der Kommandozeile

Manchmal finden sich in diesem Handbuch Verweise auf die allgemeine Subversion Dokumentation, die die Bedienung mit dem Kommandozeilen-Client (CLI) beschreibt. Um Ihnen zu verdeutlichen, welche Funktionen TortoiseSVN im Verborgenen aufruft, haben wir eine Liste der Kommandozeilenbefehle zusammengestellt, die den Funktionen der grafischen Oberfläche von TortoiseSVN entsprechen.

Anmerkung

Obwohl es äquivalente Kommandozeilenbefehle gibt, beachten Sie bitte dass TortoiseSVN *nicht* das Subversion Kommandozeilenprogramm aufruft, sondern direkt auf die Subversion Bibliotheken zugreift.

Wenn Sie einen Fehler von TortoiseSVN berichten, kann es vorkommen, dass wir Sie auffordern, dieses Fehlverhalten mit dem Kommandozeilen-Client zu reproduzieren, damit wir TortoiseSVN Probleme von Subversion Problemen unterscheiden können. Diese Kurzreferenz zeigt Ihnen, welche Befehle Sie verwenden können.

E.1. Grundregeln und Konventionen

In der folgenden Beschreibung wird die URL eines Projektarchivs einfach als URL geschrieben. Ein Beispiel wäre `http://tortoisesvn.googlecode.com/svn/trunk`. Der Pfad der Arbeitskopie wird stets als PFAD geschrieben. Ein Beispiel wäre `C:\TortoiseSVN\trunk`.



Wichtig

Da es sich bei TortoiseSVN um eine Windows Shell Erweiterung handelt, kennt es das Konzept eines Arbeitsverzeichnisses nicht. Beim entsprechenden Aufruf der Kommandozeile müssen deshalb alle Pfade zu Arbeitskopien als absolute, nicht als relative Pfade angegeben werden.

Einige Kommandozeilenbefehle besitzen Optionen. Diese werden in TortoiseSVN durch Optionsfelder gesteuert. In der Kommandozeilennotation werden diese optionalen Parameter durch [eckige Klammern] dargestellt.

E.2. TortoiseSVN Befehle

E.2.1. Auschecken

```
svn checkout [-N] [--ignore-externals] [-r rev] URL PATH
```

Wenn Nur den obersten Ordner auschecken gewählt ist, verwenden Sie die `-N` Option.

Wenn Externals auslassen gewählt ist, verwenden Sie die `--ignore-externals` Option.

Wenn Sie eine bestimmte Revision auschecken, geben Sie diese nach der URL mittels `-r` an.

E.2.2. Aktualisieren

```
svn info URL_of_WC
```

```
svn update [-r rev] PATH
```

Das Aktualisieren mehrerer Dateien ist keine atomare Operation. TortoiseSVN bestimmt zunächst die neueste Revision (HEAD) im Projektarchiv und aktualisiert anschließend alle Objekte zu dieser Revisionsnummer, damit Sie keine aus zwei Revisionen zusammengesetzte Arbeitskopie erhalten.

Wenn nur ein Objekt zur Aktualisierung ausgewählt ist oder die gewählten Objekte aus verschiedenen Projektarchiven stammen, aktualisiert TortoiseSVN zur neuesten Revision (HEAD).

An dieser Stelle werden keine Kommandozeilenoptionen verwendet. Aktualisiere zu Revision implementiert ebenfalls den Aktualisieren Befehl, bietet jedoch weitere Optionen.

E.2.3. Aktualisieren zu Revision

```
svn info URL_of_WC  
svn update [-r rev] [-N] [--ignore-externals] PATH
```

Wenn Nur den obersten Ordner auschecken gewählt ist, verwenden Sie die `-N` Option.

Wenn Externals auslassen gewählt ist, verwenden Sie die `--ignore-externals` Option.

E.2.4. Übertragen

In TortoiseSVN nutzt der Übertragen-Dialog mehrere Subversion Befehle. Der erste Schritt ist eine Statusprüfung, die die Objekte in Ihrer Arbeitskopie ermittelt, die übertragen werden können. Sie können diese Liste überprüfen, Dateien mit dem Original (BASE) vergleichen und die Objekte, die Sie übertragen wollen, auswählen.

```
svn status -v PATH
```

Wenn Zeige nicht versionierte Dateien gewählt ist, wird TortoiseSVN alle nicht versionierten Dateien und Ordner in der Arbeitskopie, unter Berücksichtigung der Ausschluss- / Ignorier-Regeln anzeigen. Diese Funktion besitzt in Subversion kein Äquivalent, da der `svn status` Befehl nicht in unversionierte Ordner hinabsteigt.

Wenn Sie nicht versionierte Dateien oder Ordner auswählen, werden diese vor dem Übertragen zur Arbeitskopie hinzugefügt.

```
svn add PATH...
```

Sobald Sie auf OK klicken, wird die Übertragung zum Projektarchiv durchgeführt. Falls Sie alle Dateiauswahlboxen in der Voreinstellung belassen haben, wird TortoiseSVN die Daten in einem Schritt rekursiv übertragen. Sobald Sie einzelne Objekte abwählen, muss eine nicht-rekursive (`-N`) Übertragung durchgeführt werden. Dabei muss jeder Pfad einzeln auf der Kommandozeile übergeben werden.

```
svn commit -m "LogMessage" [-N] [--no-unlock] PATH...
```

Die Logmeldung entspricht dem Inhalt des Eingabefeldes für Logmeldungen und kann leer gelassen werden.

Wenn Sperren behalten gewählt ist, verwenden Sie die `--no-unlock` Option.

E.2.5. Vergleich

```
svn diff PATH
```

Wenn Sie Vergleich aus dem Kontextmenü heraus aufrufen, bilden Sie die Unterschiede zwischen einer geänderten Datei und deren BASE Revision. Die Subversion Kommandozeile tut dies auch, liefert jedoch die Ausgabe in einem standard (unified) Diff-Format. TortoiseSVN hingegen verwendet TortoiseMerge (oder ein Vergleichsprogramm Ihrer Wahl), um die Unterschiede zwischen Textdateien graphisch anzuzeigen. Aus diesem Grunde gibt es keinen äquivalenten Befehl auf der Kommandozeile.

Sie können auch die Differenzen von zwei beliebigen anderen Dateien anzeigen lassen, ob diese unter Versionskontrolle stehen oder nicht. TortoiseSVN reicht die markierten Dateien einfach an das gewählte Diff-Programm weiter und lässt dieses die Änderungen ermitteln.

E.2.6. Zeige Log

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

Standardmäßig versucht TortoiseSVN 100 Logmeldungen mittels der `--limit` Option zu holen. Falls die Einstellungen jedoch den Einsatz der alten API für die Logmeldungen erfordern, wird die zweite Variante genutzt, um die Logmeldungen für 100 Revisionen im Projektarchiv zu holen.

Wenn Bei Kopien/Umbenennen anhalten gewählt ist, verwenden Sie die `--stop-on-copy` Option.

E.2.7. Prüfe auf Änderungen

```
svn status -v PATH
  or
svn status -u -v PATH
```

Die erste Statusprüfung betrachtet nur Ihre lokale Arbeitskopie. Wenn Sie auf die Projektarchiv prüfen Schaltfläche klicken, wird im Projektarchiv nachgeschaut, welche Dateien durch eine Aktualisierung betroffen wären. Dies erfordert die Verwendung der `-u` Option.

Wenn Zeige nicht versionierte Dateien gewählt ist, wird TortoiseSVN alle nicht versionierten Dateien und Ordner in der Arbeitskopie, unter Berücksichtigung der Ausschluss- / Ignorier-Regeln anzeigen. Diese Funktion besitzt in Subversion kein Äquivalent, da der `svn status` Befehl nicht in unversionierte Ordner hinabsteigt.

E.2.8. Revisionsgraph

Der Revisionsgraph ist eine Funktionalität, die nur in TortoiseSVN zur Verfügung steht und für die es kein Äquivalent in der Kommandozeile gibt.

TortoiseSVN führt die Schritte

```
svn info URL_der_AK
svn log -v URL
```

aus, wobei URL die *Basis* des Projektarchivs ist, und analysiert anschließend die zurückgegebenen Daten.

E.2.9. Projektarchivbetrachter

```
svn info URL_of_WC
svn list [-r rev] -v URL
```

Sie können `svn info` verwenden, um die Basis des Projektarchivs zu bestimmen. Dies ist die oberste, im Projektarchivbetrachter angezeigte Ebene. Sie können von dort aus nicht weiter nach Oben navigieren. Weiterhin gibt dieser Befehl alle Sperrinformationen des Projektarchivbetrachters aus.

Der `svn list` Befehl listet bei Angabe einer URL und einer Revision den Inhalt eines Verzeichnisses auf.

E.2.10. Konflikt bearbeiten

Hierfür gibt es keinen entsprechenden Kommandozeilenbefehl. Es wird TortoiseMerge oder ein externer Konflikteditor aufgerufen, um die in den Konflikt verwickelten Dateien zu betrachten und den Konflikt aufzulösen.

E.2.11. Konflikt aufgelöst

```
svn resolved PATH
```

E.2.12. Umbenennen

```
svn rename CURR_PATH NEW_PATH
```

E.2.13. Löschen

```
svn delete PATH
```

E.2.14. Rückgängig

```
svn status -v PATH
```

Der erste Schritt ist eine Statusprüfung, die die Objekte in Ihrer Arbeitskopie ermittelt, die rückgängig gemacht werden können. Sie können diese Liste überprüfen, Dateien mit dem Original (BASE) vergleichen und die Objekte, die Sie rückgängig machen wollen, auswählen.

Sobald Sie auf OK klicken, werden Ihre Änderungen rückgängig gemacht. Falls Sie alle Dateiauswahlboxen in der Voreinstellung belassen haben, führt TortoiseSVN die Aktion in einem Schritt rekursiv (-R) durch. Sobald Sie einzelne Objekte abwählen, muss jeder Pfad einzeln auf der Kommandozeile übergeben werden.

```
svn revert [-R] PATH...
```

E.2.15. Aufräumen

```
svn cleanup PATH
```

E.2.16. Sperre holen

```
svn status -v PATH
```

Der erste Schritt ist eine Statusprüfung, die die Dateien in Ihrer Arbeitskopie ermittelt, die gesperrt werden können. Sie können die Dateien, die Sie sperren wollen, auswählen.

```
svn lock -m "LockMessage" [--force] PATH...
```

Die Sperrmeldung entspricht dem Inhalt des Eingabefeldes für Logmeldungen und kann leer gelassen werden.

Wenn Sperren stehlen gewählt ist, verwenden Sie die `--force` Option.

E.2.17. Sperre freigeben

```
svn unlock PATH
```

E.2.18. Verzweigen/Markieren

```
svn copy -m "LogMessage" URL URL
  or
svn copy -m "LogMessage" URL@rev URL@rev
  or
svn copy -m "LogMessage" PATH URL
```

Der Verzweigen/Markieren-Dialog erzeugt eine Kopie im Projektarchiv. Es gibt drei Optionen:

- HEAD Revision im Projektarchiv
- Revisionsnummer im Projektarchiv
- Arbeitskopie

, die den drei Kommandozeilenparametern entsprechen.

Die Logmeldung entspricht dem Inhalt des Eingabefeldes für Logmeldungen und kann leer gelassen werden.

E.2.19. Wechseln zu

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

E.2.20. Zusammenführen

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

Die Schaltfläche Trockenlauf startet das Zusammenführen mit der `--dry-run` Option.

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

Die Standard-Diff Schaltfläche zeigt die Vergleichsoperation, die zum Zusammenführen verwendet wird.

E.2.21. Exportieren

```
svn export [-r rev] [--ignore-externals] URL Export_PATH
```

Dieser Dialog wird für nicht versionierte Ordner aufgerufen. Der Ordner ist das Ziel des Exports.

Der Export einer Arbeitskopie an einen anderen Ort wird ohne die Subversion Bibliotheken durchgeführt. Deshalb gibt es auch keinen entsprechenden Kommandozeilenbefehl.

TortoiseSVN kopiert alle Dateien an den neuen Ort und zeigt Ihnen derweil den Fortschritt an. Nicht versionierte Dateien oder Ordner können ebenfalls exportiert werden.

In beiden Fällen gilt: wenn `Externals auslassen` gewählt ist, verwenden Sie die `--ignore-externals` Option.

E.2.22. Umplatzieren

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

E.2.23. Projektarchiv hier erstellen

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

E.2.24. Hinzufügen

```
svn add PATH...
```

Wenn Sie einen Ordner markiert haben, durchsucht TortoiseSVN diesen rekursiv nach Objekten, die hinzugefügt werden können.

E.2.25. Importieren

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

Die Logmeldung entspricht dem Inhalt des Eingabefeldes für Logmeldungen und kann leer gelassen werden.

E.2.26. Annotieren

```
svn blame -r N:M -v PATH  
svn log -r N:M PATH
```

Wenn Sie die Annotierung mit TortoiseBlame betrachten, wird die Loginformation benötigt, um die Logmeldungen in einem Hinweistext anzuzeigen. Wenn Sie die Annotierungen als Textdatei betrachten, ist diese Information nicht erforderlich.

E.2.27. Ignorieren

```
svn propget svn:ignore PATH > tempfile  
{edit new ignore item into tempfile}  
svn propset svn:ignore -F tempfile PATH
```

Da die `svn:ignore` Eigenschaft oft mehrere Zeilen hat, wird hier gezeigt, wie sie mit Hilfe einer Textdatei anstatt direkt per Kommandozeile geändert wird.

E.2.28. Erzeuge Patch

```
svn diff PATH > patch-file
```

TortoiseSVN erzeugt eine Patchdatei im Standard-Diff Format, indem die Arbeitskopie mit ihrer BASE Version verglichen wird.

E.2.29. Patch anwenden

Patches anzuwenden ist ein kniffliges Unterfangen, es sei denn der Patch und die Arbeitskopie befinden sich in der gleichen Revision. Zum Glück für Sie existiert TortoiseMerge, für das es keine entsprechende Funktion in Subversion gibt.

Anhang F. Implementierungsdetails

Dieser Anhang enthält einige interne Informationen über die Arbeitsweise von TortoiseSVN.

F.1. Überlagerte Symbole

Jede Datei und jeder Ordner besitzt einen Statuswert in Subversion. Im Kommandozeilenclient werden diese durch einzelne Buchstaben gekennzeichnet, in TortoiseSVN durch überlagerte Symbole. Da die Anzahl der zur Verfügung stehenden überlagerten Symbole durch Windows eingeschränkt wird, kann ein Symbol für mehrere Statuswerte stehen.



Das *Konflikt* Symbol wird verwendet, um anzuzeigen, dass beim *Aktualisieren* oder *Wechseln* zu ein Konflikt zwischen lokalen Änderungen und Änderungen in der Arbeitskopie auftrat. Weiterhin wird durch dieses Symbol der *Versperrt* Status angezeigt.



Sobald Sie eine Datei ändern, ändert sich auch der Status der Datei auf *Verändert*. Mit diesem Symbol werden auch *Zusammengeführte* oder *Ersetzte* Dateien gekennzeichnet. Auf diese Weise können Sie mit einem Blick feststellen, welche Dateien Sie geändert und noch nicht in das Projektarchiv übertragen haben.



Dieses überlagerte Symbol zeigt, dass Dateien oder Ordner zum *Löschen* aus der Versionskontrolle markiert wurden oder dass TortoiseSVN eine Datei unter Versionskontrolle vermisst. Eine fehlende Datei kann natürlich dieses Symbol nicht erhalten, aber der Elternordner zeigt es an, sobald ein Unterobjekt vermisst wird.



Dieses Symbol zeigt an, dass eine Datei oder ein Ordner neu zur Versionskontrolle *Hinzugefügt* wurde.



Dieses Symbol bedeutet dass der Subversion Status *Normal* ist oder dass es sich um ein versioniertes Objekt handelt, dessen Status noch nicht bekannt ist. Da TortoiseSVN im Hintergrund einen Statuspeicher verwendet, kann es ein paar Sekunden dauern, bis das Symbol aktualisiert wird.



Das *Benötigt Sperre* Symbol wird verwendet, um anzuzeigen, dass bei einer Datei die `svn:needs-lock` Eigenschaft gesetzt ist. Bei Arbeitskopien, die mit Subversion 1.4.0 oder neuer erstellt wurden, wird der `svn:needs-lock` Status lokal zwischengespeichert und dazu verwendet dieses Symbol anzuzeigen. Bei Arbeitskopien, die mit einer älteren Version erstellt wurden, zeigt TortoiseSVN dieses Symbol, wenn die Datei schreibgeschützt ist. Beachten Sie, dass Subversion das Format seiner Arbeitskopien beim Aktualisieren automatisch auf den neuesten Stand bringt. Das Speichern der `svn:needs-lock` Eigenschaft erfolgt jedoch erst, wenn die Datei selbst aktualisiert wird.



Das *Gesperrt* Symbol zeigt an, dass die lokale Arbeitskopie die Sperre für die Datei besitzt.



Dieses Symbol bedeutet dass dieses Objekt von Subversion *Ignoriert* wird. Das kann aufgrund eines globalen Ignoriermusters oder der `svn:ignore` Eigenschaft des Elternobjekts der Fall sein. Dieses Symbol ist optional.



Dieses Symbol zeigt an, dass eine Datei oder ein Ordner *Nicht Versioniert* ist. Das Objekt liegt innerhalb eines versionierten Ordners, befindet sich jedoch selbst nicht unter Versionskontrolle. Dieses Symbol ist optional.

Wenn ein Objekt sich nicht in einer Arbeitskopie befindet, wird kein überlagertes Symbol angezeigt. Wenn Sie die *Ignoriert* und *Nicht Versioniert* Symbole abgeschaltet haben, wird für solche Objekte auch innerhalb einer Arbeitskopie kein Symbol angezeigt.

Ein Objekt kann nur einen Subversion Status besitzen. Zum Beispiel kann eine Datei lokal verändert und gleichzeitig zum Löschen markiert sein. Subversion gibt in diesem Fall nur einen Wert - *gelöscht* zurück. Diese Prioritäten sind in Subversion selbst festgelegt.

Wenn TortoiseSVN den Status rekursiv anzeigt (die Standardeinstellung), erhält jeder Ordner ein Symbol, das seinen eigenen sowie die Status aller seiner Kinder anzeigt. Damit ein *Summensymbol* angezeigt werden kann, verwenden wir die obenstehende Reihenfolge, um festzulegen, welches Symbol verwendet wird, wobei *Konflikt* die höchste Priorität hat.

Sie werden möglicherweise feststellen, dass nicht alle diese Symbole auf Ihrem Rechner dargestellt werden. Das liegt daran, dass Windows die Anzahl der überlagerten Symbole beschränkt. Windows selber verwendet einige, und wenn Sie gleichzeitig TortoiseCVS installiert haben, sind nicht genügend Plätze für die Symbole beider Anwendungen frei. Deshalb versucht TortoiseSVN ein „Guter Bürger(tm)“ zu sein und schränkt seine Verwendung von überlagerten Symbolen ein, damit andere Anwendungen eine Chance haben.

- *Normal*, *Verändert* und *Konflikt* sind immer geladen und sichtbar.
- *Gelöscht* wird, wenn möglich geladen, fällt aber auf *Verändert* zurück, wenn nicht genügend Plätze frei sind.
- *Schreibgeschützt* wird, wenn möglich geladen, fällt aber auf *Verändert* zurück, wenn nicht genügend Plätze frei sind.
- Das *Gesperrt* Symbol wird nur geladen, wenn weniger als 13 Plätze belegt sind. Es fällt auf das *Normal* Symbol zurück, wenn nicht genügend frei sind.
- Das *Hinzugefügt* Symbol wird nur geladen, wenn weniger als 14 Plätze belegt sind. Es fällt auf das *Verändert* Symbol zurück, wenn nicht genügend frei sind.

Anhang G. Svnserve mit SSH sichern

Dieser Abschnitt bietet eine schrittweise Anleitung wie man Subversion und TortoiseSVN so einrichtet, dass sie das `svn+ssh` Protokoll verwenden. Falls Sie bereits ein SSH Anmeldeverfahren für Ihren Server nutzen, sind schon alle Voraussetzungen geschaffen und Sie finden weitere Details im Subversion Buch. Wenn Sie SSH noch nicht verwenden, dies aber möchten, um Ihre Subversion Installation abzusichern, bietet Ihnen diese Anleitung ein einfaches Verfahren, das kein separates SSH Konto für jeden Nutzer auf dem Server erfordert.

In dieser Implementierung legen wir ein einzelnes SSH Konto für alle Subversion Nutzer an und verwenden unterschiedliche Anmeldeschlüssel zur Unterscheidung der einzelnen Anwender.

In diesem Anhang setzen wir voraus, dass Sie bereits Subversion installiert und, wie anderweitig im Handbuch beschrieben, ein Projektarchiv angelegt haben. Beachten Sie bitte, dass Sie bei Verwendung von SSH `svnserve` *nicht* als Dämon oder Dienst starten sollten.

Ein Großteil der folgenden Informationen stammt aus einer Anleitung von Marc Logemann, die sich unter www.logemann.org [<http://www.logemann.org/2007/03/13/subversion-tortoisesvn-ssh-howto/>] findet. Weitere Hinweise zum Einrichten eines Windows Servers stammen von Thorsten Müller. Danke Jungs!

G.1. Einen Linux Server einrichten

Sie müssen SSH auf dem Server aktiviert haben. Wir setzen hier voraus, dass Sie OpenSSH verwenden. Bei den meisten Distributionen ist OpenSSH bereits installiert. Zum Überprüfen geben Sie:

```
ps xa | grep sshd
```

ein und suchen Sie nach SSH Jobs.

Bitte beachten Sie folgendes: Wenn Sie Subversion selber aus den Quellen erstellen und keine Parameter an `./configure` übergeben, wird Subversion ein `bin` Verzeichnis unter `/usr/local` erstellen und seine Binärdateien dort ablegen. Falls Sie SSH Tunnel verwenden wollen, müssen Sie sich bewusst sein, dass der Benutzer, der sich per SSH anmeldet, `svnserve` und einige andere Binärdateien ausführen können muss. Aus diesem Grund sollten Sie entweder `/usr/local/bin` in die `PATH` Variable aufnehmen oder symbolische Verweise Ihrer Binärdateien in `/usr/sbin` oder einem anderen Verzeichnis erstellen, das normalerweise in der `PATH` Variable steht.

Um sicherzustellen, dass alles OK ist, melden Sie sich als der gewünschte Anwender per SSH an und geben Sie:

```
which svnserve
```

ein. Dieser Befehl sagt Ihnen, ob `svnserve` erreichbar ist.

Legen Sie einen neuen Benutzer an, mit dem wir auf das SVN Projektarchiv zugreifen:

```
useradd -m svnuser
```

Stellen Sie sicher, dass dieser Anwender volle Zugriffsrechte auf das Projektarchiv hat.

G.2. Einen Windows Server einrichten

Installieren Sie den Cygwin SSH Dienst, so wie in der folgenden Anleitung beschrieben: <http://pigtail.net/LRP/printsrv/cygwin-sshd.html>

Legen Sie ein neues Benutzerkonto namens `svnuser` unter Windows an, mit dem wir auf das Projektarchiv zugreifen wollen. Stellen Sie sicher, dass dieses Konto volle Zugriffsrechte auf das Projektarchiv hat.

Falls es noch keine Passwortdatei gibt, legen Sie von der Cygwin Konsole aus mittels:

```
mkpasswd -l > /etc/passwd
```

eine an.

G.3. SSH Werkzeuge zur Verwendung mit TortoiseSVN

Die Programme, die wir zur Einrichtung von SSH auf der Windows Seite benötigen, finden sich auf der folgenden Webseite: <http://www.chiark.greenend.org.uk/~sgtatham/putty/> Gehen Sie in den Download Bereich und laden Sie Putty, Plink, Pageant und Puttygen herunter.

G.4. Erzeugen von OpenSSH Zertifikaten

Der nächste Schritt ist das Anlegen eines Schlüsselpaares für die Anmeldung. Es gibt zwei Möglichkeiten, diese zu erzeugen. In der ersten Variante erstellen Sie die Schlüssel mittels PuTTYgen auf dem Client, laden den öffentlichen Schlüssel auf den Server und verwenden den privaten Schlüssel mit PuTTY. In der anderen erstellen Sie das Schlüsselpaar mit dem OpenSSH Programm ssh-keygen, laden den privaten Schlüssel auf Ihren Client herunter und konvertieren diesen in einen privaten Schlüssel im PuTTY Stil.

G.4.1. Schlüssel mittels ssh-keygen erzeugen

Melden Sie sich als root oder svnuser am Server an und geben Sie:

```
ssh-keygen -b 1024 -t dsa -N Passphrase -f Schlüsseldatei
```

ein, wobei sie eine echte Passphrase (die nur Sie selber kennen) und die Schlüsseldatei angeben. Jetzt haben Sie einen 1024 Bit SSH2 DSA Schlüssel erzeugt. Wenn Sie

```
ls -l Schlüsseldatei*
```

eingeben, sehen Sie zwei Dateien, Schlüsseldatei und Schlüsseldatei.pub. Wie Sie schon erraten haben, ist die .pub Datei der öffentliche Schlüssel und die andere der private.

Hängen Sie den öffentlichen Schlüssel an die Liste in dem .ssh Ordner innerhalb des svnuser Home-Verzeichnis an:

```
cat Schlüsseldatei.pub >> /home/svnuser/.ssh/authorized_keys
```

Damit wir den auf diese Weise erzeugten privaten Schlüssel verwenden können, muss er in das PuTTY Format konvertiert werden. Das liegt daran, dass das Format für private Schlüssel nirgendwo standardisiert ist. Nachdem Sie die private Schlüsseldatei auf Ihren PC heruntergeladen haben, starten Sie PuTTYgen und rufen Sie Conversions → Import key auf. Navigieren Sie zu Ihrer vom Server heruntergeladenen Schlüsseldatei und geben Sie die Passphrase ein, die sie beim Erzeugen des Schlüssels gewählt haben. Schließlich klicken Sie auf Save private key und speichern Die Datei als Schlüsseldatei.PPK

G.4.2. Schlüssel mittels PuTTYgen erzeugen

Benutzen Sie PuTTYgen, um ein Schlüsselpaar zu generieren und abzuspeichern. Kopieren Sie den öffentlichen Schlüssel auf den Server und hängen Sie ihn an die Liste in dem .ssh Ordner innerhalb des svnuser Home-Verzeichnis an:

```
cat Schlüsseldatei.pub >> /home/svnuser/.ssh/authorized_keys
```

G.5. Test mittels PuTTY

Zum Test der Verbindung verwenden wir PuTTY. Starten Sie die Anwendung, wählen Sie die Session (Sitzung) Seite und tragen Sie dort im Feld Hostname den Namen oder die Adresse Ihres Servers ein. Wählen Sie das SSH Protokoll und speichern Sie die Einstellungen als SvnVerbindung oder was auch

immer Sie bevorzugen. Auf der **SSH** Seite stellen Sie die bevorzugte SSH Protokollversion auf 2 und in der **Auth** Seite tragen Sie den vollständigen Pfad zu der privaten Schlüsseldatei `.PPK` ein, die sie vorher konvertiert haben. Gehen Sie zurück zur **Sessions** Seite und speichern (**Save**) Sie die Einstellungen. Sie sehen nun `SvnVerbindung` in der Liste der gespeicherten Sitzungen.

Klicken Sie auf **Open** und Sie sollten ein Telnet-ähnliches Anmeldefenster sehen. Geben Sie `svnuser` als Benutzernamen ein und wenn alles stimmt, sollten Sie direkt, ohne weiter Passwortabfrage, verbunden werden.

Sie müssen unter Umständen die `/etc/sshd_config` Datei auf dem Server anpassen. Ändern Sie die Zeilen wie unten angegeben und starten Sie den SSH Dienst anschließend neu

```
PubkeyAuthentication yes
PasswordAuthentication no
PermitEmptyPasswords no
ChallengeResponseAuthentication no
```

G.6. SSH mit TortoiseSVN testen

Bisher haben wir nur überprüft, ob Sie sich über SSH anmelden können. Nun müssen wir sicherstellen, dass die SSH Verbindung tatsächlich `svnserve` starten kann. Dazu müssen Sie auf dem Server die Datei `/home/svnuser/.ssh/authorized_keys` so anpassen, dass viele Autoren dasselbe Systemkonto, `svnuser` benutzen können. Beachten Sie bitte, dass jeder Autor die gleiche Anmeldung aber einen unterschiedlichen Anmeldeschlüssel verwendet, so dass Sie für jeden Autor eine eigene Zeile eintragen müssen.

Achtung: Dies steht alles auf einer sehr langen Zeile:

```
command="svnserve -t -r <ReposRootPath> --tunnel-user=<author>",
no-port-forwarding,no-agent-forwarding,no-X11-forwarding,
no-pty ssh-rsa <PublicKey> <Comment>
```

Es gibt mehrere Werte, die Sie an Ihre Einstellungen anpassen müssen.

`<ReposRootPath>` sollte durch den Pfad zu dem Verzeichnis, das Ihre Projektarchive enthält, ersetzt werden. Dadurch vermeiden Sie, dass vollständige Serverpfade innerhalb der URLs angegeben werden müssen. Beachten Sie, dass Sie selbst auf einem Windows Server normale Schrägstriche zum Trennen der Pfade verwenden müssen, z.B. `c:/svn/reposroot`. Im untenstehenden Beispiel nehmen wir an, dass sich die Projektarchive in einem Basisordner namens `repos` befinden.

`<author>` sollte durch den SVN Autor ersetzt werden, der in das Log eingetragen werden soll. Dies ermöglicht es `svnserve`, seine eigenen Berechtigungen innerhalb von `svnserve.conf` zu verwenden.

`<PublicKey>` sollte durch den bereits von Ihnen erzeugten öffentlichen Schlüssel ersetzt werden.

`<Comment>` kann ein beliebiger Kommentar sein, sollte aber sinnvollerweise eine Abbildung des Autoren auf den richtigen Namen der Person sein.

Machen Sie einen Rechtsklick auf einen beliebigen Ordner im Windows Explorer und wählen Sie **TortoiseSVN** → **Projektarchiv**. Sie werden nach einer URL gefragt, die Sie in der Form:

```
svn+ssh://svnuser@SvnVerbindung/repos
```

eingeben. Was bedeutet diese URL? Der Name des Schemas lautet `svn+ssh` wodurch TortoiseSVN weiß, wie Anfragen an den Server behandelt werden. Nach dem Doppelschrägstrich geben Sie den Anwender an, der sich mit dem Server verbindet, in unserem Fall `svnuser`. Hinter dem `@` geben wir unseren PuTTY Sitzungsnamen an. Diese Sitzung enthält alle Details, z.B. wo sich der private Schlüssel befindet sowie die IP Adresse des Servers oder sein Name. Schließlich folgt noch der Pfad zum Projektarchiv relativ zum Basisverzeichnis auf dem Server, wie es in der `authorized_keys` Datei festgelegt wurde.

Klicken Sie auf OK und Sie sollten den Inhalt des Projektarchivs durchblättern können. Wenn ja, haben Sie jetzt einen funktionierenden SSH Tunnel mit TortoiseSVN.

Beachten Sie bitte, dass normalerweise TortoiseSVN beim Verbindungsaufbau eine eigene Version von Plink verwendet. Diese Spezialversion zeigt bei einem Anmeldeversuch kein Konsolenfenster an. Das bedeutet aber auch, dass es keine Ausgabemöglichkeit für Fehlermeldungen gibt. Falls sie die Fehlermeldung „Unable to write to standard output“ bekommen, sollten Sie Plink als Client in den TortoiseSVN Netzwerkeinstellungen festlegen. Dadurch können Sie die tatsächlich von Plink generierten Fehlermeldungen sehen.

G.7. SSH Konfigurationsvarianten

Eine Möglichkeit, die URL in TortoiseSVN zu vereinfachen, besteht darin, den Benutzer innerhalb der PuTTY Sitzung festzulegen. Dazu laden Sie Ihre bereits definierte `SvnVerbindung` in PuTTY und tragen auf der Connection Seite unter Auto login user den Benutzernamen ein, z.B. `svnuser`. Speichern Sie die PuTTY Sitzung und testen Sie folgende URL innerhalb von TortoiseSVN

```
svn+ssh://SvnVerbindung/repos
```

Dieses Mal übergeben wir nur die PuTTY Sitzung `SvnConnection` an den von TortoiseSVN verwendeten SSH Client (TortoisePlink.exe). Der Client lädt alle weiteren Details aus der Sitzung.

Derzeit prüft PuTTY nicht alle gespeicherten Konfigurationen. Falls Sie mehrere Konfigurationen mit demselben Servernamen haben, wählt PuTTY die erste passende. Außerdem wird beim Ändern und Speichern der Standardkonfiguration der Benutzername *nicht* mit gespeichert.

Viele Leute benutzen Pageant zum Speichern ihrer privaten Schlüssel. Da eine PuTTY Sitzung im Stande ist, einen Schlüssel zu speichern, benötigen Sie Pageant nicht unbedingt. Sollten Sie aber Schlüssel für verschiedene Server benötigen, müssten Sie die PuTTY Sitzung für jeden Server ändern. In dieser Situation ist der Einsatz von Pageant sinnvoll, denn wenn PuTTY, Plink, TortoisePlink oder ein anderes PuTTY-basiertes Programm sich mit einem SSH Server verbindet, prüft es beim Verbindungsaufbau alle privaten Schlüssel, die Pageant bereitstellt.

Zu diesem Zweck starten Sie Pageant und fügen Sie den privaten Schlüssel hinzu. Es sollte derselbe Schlüssel wie in der oben definierten PuTTY Sitzung sein. Falls Sie Pageant benutzen, können Sie die Verweise auf die private Schlüsseldatei aus der PuTTY Sitzung löschen. Sie können weitere Schlüssel für andere Server oder Benutzer hinzufügen.

Wenn Sie diese Prozedur nicht bei jedem Neustart Ihres PCs wiederholen wollen, sollten Sie Pageant zur Autostart Gruppe von Windows hinzufügen. Sie können die Schlüssel mit vollständigen Pfaden als Kommandozeilenparameter an Pageant übergeben.

Die letzte Möglichkeit, sich mit einem SSH Server zu verbinden besteht einfach darin, diese URL innerhalb von TortoiseSVN anzugeben:

```
svn+ssh://svnuser@100.101.102.103/repos  
svn+ssh://svnuser@mydomain.com/repos
```

Wie Sie sehen verwenden wir keine gespeicherte PuTTY Sitzung sondern eine IP Adresse (oder einen Domänennamen) als Verbindungsziel. Wir geben auch den Benutzer an, aber Sie fragen sich sicherlich wie die private Schlüsseldatei gefunden wird. Da TortoisePlink nur eine abgewandelte Version des Standard Plink aus dem PuTTY Paket ist, verwendet TortoiseSVN alle in Pageant abgespeicherten Schlüssel.

Falls Sie die letztgenannte Methode verwenden, stellen Sie sicher, dass Sie keinen Standardbenutzernamen in PuTTY eingetragen haben. Es gibt Berichte über einen Fehler in PuTTY, der dazu führt, dass die Verbindung in diesem Fall geschlossen wird. Um den Standardbenutzer zu entfernen, löschen Sie einfach `HKEY_CURRENT_USER\Software\SimonTatham\Putty\Sessions\Default%20Settings\HostName`

Glossar

Aktualisieren	Dieser Subversionbefehl holt die neuesten Änderungen aus dem Projektarchiv in Ihre Arbeitskopie und führt dabei die Änderungen von anderen mit Ihren Änderungen zusammen.
Annotieren	Dieser Befehl steht nur für Textdateien zur Verfügung und er annotiert jede Zeile mit der Revision und dem letzten Autor. In unserer GUI Anwendung namens TortoiseBlame wird zusätzlich die Logmeldung angezeigt, wenn Sie mit der Maus über die Revisionsnummer fahren.
Arbeitskopie	Dies ist Ihr lokaler „Sandkasten“, der Bereich, in dem Sie an versionierten Dateien arbeiten, und er liegt normalerweise auf Ihrer lokalen Festplatte. Sie erzeugen eine Arbeitskopie, indem Sie einen Ordner aus einem Projektarchiv „Auschecken“ und Sie führen Ihre Änderungen mit einem „Übertragen“ in das Projektarchiv zurück.
Aufräumen	Ein Zitat aus dem Subversion Buch: „Räumt die Arbeitskopie rekursiv auf, entfernt dabei Sperren und setzt unvollendete Operationen fort. Wenn Sie jemals einen <i>Arbeitskopie gesperrt</i> Fehler bekommen, führen Sie diesen Befehl aus, um veraltete Sperren zu entfernen und Ihre Arbeitskopie wieder in einen nutzbaren Zustand zu versetzen.“ Beachten Sie, dass in diesem Zusammenhang <i>Sperren</i> lokale Dateisystemsperrern bezeichnet und keine Sperren im Projektarchiv.
Auschecken	Ein Subversionbefehl, der eine lokale Arbeitskopie in einem leeren Verzeichnis erstellt, indem versionierte Dateien aus einem Projektarchiv heruntergeladen werden.
BASE Revision	Die aktuelle Basisrevision einer Datei oder eines Ordners in Ihrer <i>Arbeitskopie</i> . Dies ist die Revision in der sich die Datei oder der Ordners befand als zuletzt Auschecken, Aktualisieren oder Übertragen aufgerufen wurde. Die BASE Revision entspricht normalerweise nicht der HEAD Revision
BDB	Berkeley DB. Ein bewährtes Datenbanksystem für Projektarchive. Kann nicht über Netzwerkfreigaben genutzt werden. Standard für Projektarchive von Subversion vor Version 1.2.
Eigenschaft	Zusätzlich zur Versionierung von Dateien und Ordnern erlaubt Subversion die Versionierung von Metadaten für Ordner und Dateien, auch als „Eigenschaften“ bezeichnet. Jede Eigenschaft hat einen Namen und einen Wert, ähnlich wie ein Registrierungsschlüssel. Subversion selbst hat einige spezielle Eigenschaften definiert, welche intern benutzt werden, wie zum Beispiel <code>svn:eol-style</code> . TortoiseSVN nutzt ebenfalls spezielle Eigenschaften, so zum Beispiel <code>tsvn:logminsize</code> . Sie können eigene Eigenschaften mit jedem Namen und Wert erstellen den sie möchten.
Exportieren	Dieser Befehl erzeugt eine Kopie eines versionierten Ordners. Wie eine Arbeitskopie jedoch ohne die lokalen <code>.svn</code> Verzeichnisse.
FSFS	Ein proprietäres Subversion-Dateisystem für Projektarchive. Kann über Netzwerkfreigaben genutzt werden. Seit Version 1.2 Standard für neue Projektarchive.

GPO	Gruppenrichtlinienobjekt
HEAD Revision	Die neueste Version einer Datei oder eines Ordners im <i>Projektarchiv</i> .
Hinzufügen	Ein Subversion Befehl, der zum Hinzufügen von Dateien oder Ordnern zu einem Projektarchiv benutzt wird. Die neuen Objekte werden beim Übertragen zum Projektarchiv hinzugefügt.
Historie	Zeigt die Revisionshistorie einer Datei oder eines Ordners. Auch als „Log“ bekannt.
Importieren	Ein Subversionbefehl mit dem eine vollständige Ordnerhierarchie mit einer einzigen Revision in ein Projektarchiv importiert wird.
Konflikt	Wenn Änderungen vom Projektarchiv mit den lokalen Änderungen zusammengeführt werden sollen, kann es vorkommen dass diese Änderungen in den selben Zeilen der Datei auftreten. In diesem Fall kann Subversion nicht selbst entscheiden welche Version zu verwenden ist und die Datei wird somit als „in Konflikt“ markiert. Sie müssen die Datei von Hand editieren und den Konflikt auflösen, bevor Sie weitere Änderungen übertragen können.
Konflikt lösen	Wenn sich Dateien in einer Arbeitskopie nach einem Zusammenführen in einem Konflikt befinden, müssen diese Konflikte von einer Person mittels eines Editors (oder vielleicht TortoiseMerge) aufgelöst werden. Dieser Prozess wird als „Konflikt lösen“ bezeichnet. Wenn die Konflikte aufgelöst sind, können diese Dateien als aufgelöst markiert werden, was eine Übertragung erlaubt.
Kopieren	In einem Subversion Projektarchiv können Sie eine Kopie einer einzelnen Datei oder eines ganzen Baumes erstellen. Diese sind als „billige Kopien“ in Form eines Verweises auf das Original implementiert. Somit benötigt selbst die Kopie eines Dateibaumes fast keinen Platz. Eine Kopie behält die Historie ihrer Vorgänger, so dass auch Logmeldungen über die Kopie hinaus verfolgt werden können.
Löschen	Wenn Sie ein versioniertes Objekt löschen (und dann Übertragen), existiert die Datei oder Ordner nach der übertragenen Revision nicht mehr im Projektarchiv. Aber selbstverständlich existiert diese noch in früheren Revisionen und es kann immer noch darauf zugegriffen werden. Falls notwendig, können Sie das gelöschte Objekt wieder in die Arbeitskopie kopieren und diese so mit der kompletten Historie „wieder herstellen“.
Patch	Wenn eine Arbeitskopie nur Änderungen an Textdateien aufweist, ist es möglich mit dem Diff-Befehl von Subversion eine Datei zu erstellen, welche diese Änderungen im Standard-Diff Format enthält. Eine Datei mit solchem Inhalt wird üblicherweise als „Patch“ bezeichnet und kann zum Beispiel jemand anderem per Mail geschickt werden, der diese Änderungen dann in seine Arbeitskopie übernehmen kann. Jemand ohne Übertragungsrechte zum Projektarchiv kann so Änderungen vornehmen, diese Änderungen an eine Person mit Übertragungsrechten schicken, welche dann diese Änderungen übertragen kann. Oder man kann eine solche Patchdatei anderen zuerst zur Diskussion schicken, bevor man die Änderungen überträgt.
Projektarchiv	Ein Projektarchiv ist ein zentraler Platz in dem Daten gespeichert und verwaltet werden. Das Projektarchiv kann sowohl auf einem Server

	liegen und über das Netzwerk angesprochen werden, als auch auf der lokalen Platte liegen.
Protokoll	Zeigt das Änderungsprotokoll einer Datei oder eines Ordners. Auch „Historie“ genannt.
Revision	<p>Jedes Mal, wenn Sie Änderungen übertragen, wird eine neue „Revision“ im Projektarchiv erstellt. Jede Revision stellt den Status des Projektarchivs zu einem bestimmten Zeitpunkt in der Vergangenheit dar. Wenn Sie in der Zeit zurückgehen möchten, können Sie den Zustand des Projektarchivs, wie es zur Revision N war, ansehen.</p> <p>Anders ausgedrückt bezieht sich eine Revision auf eine Menge von Änderungen, die, als diese Revision erstellt wurde, vorgenommen wurden.</p>
Revisionseigenschaft (revprop)	So wie Dateien Eigenschaften haben können, hat auch jede Revision im Projektarchiv Eigenschaften. Einige spezielle Revisionseigenschaften werden automatisch hinzugefügt, wenn die Revision erstellt wird. Diese sind: <code>svn:date</code> , <code>svn:author</code> <code>svn:log</code> welche das Übertragungsdatum, die Person welche die Übertragung gemacht hat und die Logmeldung darstellen. Diese Eigenschaften können verändert werden, aber sie sind nicht versioniert, was bedeutet dass jede Änderung permanent ist und nicht rückgängig gemacht werden kann.
Rückgängig	Subversion hält eine lokale „Basis“-Kopie von jeder Datei in dem Zustand, in dem sie zuletzt aktualisiert wurde in der Arbeitskopie. Wenn Sie Änderungen gemacht haben und diese rückgängig machen wollen, können Sie den Befehl „Rückgängig“ verwenden, um zurück zur Basisversion der Datei zu kommen.
Sperren	Wenn Sie eine Datei sperren, wird diese im Projektarchiv als nicht übertragbar markiert, mit Ausnahme der Arbeitskopie von der aus die Sperre gesetzt wurde.
SVN	<p>Eine häufig verwendete Abkürzung für Subversion</p> <p>Der Name des spezifischen Protokolls, das von „svnserve“ genutzt wird.</p>
Übertragen	Dieser Subversionbefehl wird verwendet, um die Änderungen in Ihrer lokalen Arbeitskopie zurück in das Projektarchiv zu übertragen.
Umplatzieren	<p>Wenn das Projektarchiv verschoben wird, vielleicht weil Sie es in einen anderen Ordner auf dem Server verschoben haben oder der Domänenname des Servers sich ändert, dann müssen Sie die Arbeitskopie „umplatzieren“, so dass die Projektarchiv-URL zum neuen Ort zeigt.</p> <p>Hinweis: Sie sollten diesen Befehl nur benutzen wenn die Arbeitskopie zur selben Stelle im Projektarchiv selbst zeigt und nur das Projektarchiv selbst verschoben wurde. In allen anderen Situationen müssen Sie den Befehl „Wechseln zu“ verwenden.</p>
Vergleich	Abkürzung für „Zeige Unterschiede“. Sehr nützlich, wenn Sie genau sehen wollen, welche Änderungen Sie vorgenommen haben.
Verzweigen	Ein Begriff der häufig in Versionskontrollsystemen verwendet wird, um zu beschreiben, wenn sich die Entwicklung an einem bestimmten

Punkt verzweigt und verschiedenen Wegen folgt. Sie können von der Hauptentwicklungslinie abzweigen, um neue Funktionen zu implementieren ohne den Hauptzweig instabil zu machen. Genauso können Sie eine stabile, freigegebene Version abzweigen, an der Sie nur noch Fehler beseitigen, während die Weiterentwicklung am instabilen Hauptzweig erfolgt. In Subversion sind Zweige als „billige Kopien“ implementiert.

Wechseln zu

So wie „Aktualisieren zu Revision“ den Zeitpunkt der Arbeitskopie ändert, um auf einen anderen Punkt in der Vergangenheit zu zeigen, ändert „Wechseln zu“ die Position der Arbeitskopie, so dass sie an einen anderen Ort im Projektarchiv zeigt. Dies ist vor allem nützlich, wenn Sie mit verschiedenen Zweigen arbeiten, in denen nur wenige Dateien unterschiedlich sind. Sie können ihre Arbeitskopie zwischen den Zweigen hin und her wechseln wobei jeweils nur die Unterschiede übertragen werden.

Zusammenführen

Der Prozess mit welchem Änderungen vom Projektarchiv zur Arbeitskopie hinzugefügt werden, ohne lokale Änderungen zu beeinträchtigen. Manchmal können diese Änderungen nicht automatisch hinzugefügt werden und die Arbeitskopie befindet sich dann in einem Konflikt.

Das Zusammenführen passiert automatisch, wenn Sie ihre Arbeitskopie aktualisieren. Sie können auch mit dem TortoiseSVN Befehl „Zusammenführen“ spezifische Änderungen von einem Zweig in der Arbeitskopie zusammenführen.

Stichwortverzeichnis

Symbole

.svn Ordner, 165
_svn Ordner, 165

A

Aktionsskripte, 19, 159
aktualisieren, 52, 179
Änderung rückgängig machen, 179
Änderungen, 60, 180
Änderungen anzeigen, 58
Änderungen exportieren, 79
Änderungen holen, 52
Änderungen übertragen, 46
Änderungsliste, 62
Anmeldung, 40
Annotieren, 118
Apache, 26
Arbeitskopie, 9
Arbeitskopie erstellen, 44
ASP Projekte, 185
Auf neue Version prüfen, 184
Aufräumen, 92
Auscheck Verweis, 19
Auschecken, 44
ausliefern, 184
Auslieferung markieren, 101
ausschließen, 137
auto-props, 96
Automatisierung, 187, 190
Autor bearbeiten, 72
Autorisierung, 30

B

Backup, 18
Baumkonflikt, 54
Bildvergleich, 81
blame, 118
bugtracker, 131, 131

C

CLI, 192
COM, 167, 172
COM Schnittstelle von SubWCRev, 169

D

Dateien kopieren, 83
Dateien umbenennen, 83
Dateien vergleichen, 180
Dateien verschieben, 83
Dateien zum Projektarchiv hinzufügen, 41
Dateimuster, 86
Datenintegration, 111
Domänencontroller, 31, 184

Drag and Drop, 39
Drag und Drop, 39

E

Eigenschaften, 92
Einchecken, 46
Einstellungen, 136
Entfernen, 87
entversionieren, 130, 182
Erneut zusammenführen, 112
Erstellen
 Kommandozeilen-Client, 15
 TortoiseSVN, 15
Explorer, 1
Explorer Spalten, 60
Exportieren, 128
externals, 99, 180
Externe Projektarchive, 99

F

FAQ, 177
Fehlerverfolgung, 131
Filter, 73
Funktionen deaktivieren, 185

G

Gemeinsame Projekte, 180
global ignore, 137
GPO, 184
Graph, 123
Groß-/Kleinschreibung, 89
Gruppenrichtlinien, 184, 185

H

Herstellerprojekte, 180
hinzufügen, 82
Historie, 64
hooks, 19

I

IBugtraqProvider, 172
Ignorieren, 84
import, 41
Import an Ort und Stelle, 43
Index der Projekte, 30
Installation, 3
issuetracker, 131, 172

K

Kommandozeile, 187, 190
Kommandozeilen-Client, 192
Kommentare, 64
Konflikt, 9, 54
Konflikt lösen, 54
Konflikte, 111
Konflikteditoren, 82
Kontextmenü, 37

Kontextmenüeinträge, 185
Kopie, 101, 121

L

Leere Logmeldung, 178
locking, 113
Log, 64
Log-Puffer, 156
Logmeldung, 178, 178
Logmeldungen, 64
lokale Aktionsskripte, 159
Löschen, 87

M

markieren, 83, 101
maximieren, 41
Mehrfachanmeldung, 32
Merge, 104
 Einen Revisionsbereich, 106
 Erneut zusammenführen, 107
 Zwei Zweige, 108
Modul, 172
mod_authz_svn, 27, 30
msi, 184

N

Netzwerkfreigabe, 16
Neue Dateien versionieren, 82
Neuer Server, 130
nicht versionierte Dateien/Ordner, 84
NTLM, 31

O

Ordner vergleichen, 180

P

patch, 117
Platzhalter, 86
praise, 118
Priorität der überlagerten Symbole, 198
Projektarchiv, 5, 5, 41
Projektarchiv anlegen, 15
Projektarchiv URL geändert, 130
Projektarchivbetrachter, 121, 135
Projekteigenschaften, 97
Protokoll der Datenintegration, 71
Proxy Server, 150

R

Rechts-Ziehen, 39
Rechtschreibprüfung, 3
Rechtsklick, 37
Registrierung, 164
Revision, 12, 123
Revisionen vergleichen, 79
Revisionseigenschaften, 72
Revisionsgraph, 123

Revisionsnummern im Quelltext, 167
revprops, 72
Rückgängig, 90, 90, 179, 179

S

SASL, 24
Schlüsselwörter, 94
Schlüsselwörter expandieren, 94
schreibgeschützt, 113
SCM, 1
Server umgezogen, 130
Serverbetrachter, 121
Serverseitige Aktionen, 121
Serverseitige Aktionsskripte, 19
sounds, 136
spezielle Dateien, 43
Sprachpakete, 3
SSL, 33
SSPI, 31
Standard-Diff, 117
Statistiken, 74
Status, 58, 60
Status der Arbeitskopie, 58
SUBST Laufwerke, 149
Subversion Buch, 5
Subversion Eigenschaften, 93
SubWCRev, 167
SVNParentPath, 28, 30
SVNPath, 28
svnserve, 20, 22
SVN_ASP_DOT_NET_HACK, 185
switch, 103
Symbole, 58, 58, 198

T

temporäre Dateien, 41
TortoiseDiff, 81
TortoiseSVN Eigenschaften, 97
TortoiseSVN Verweis, 19

U

Übersetzungen, 3
übertragen, 46
Übertragen rückgängig machen, 179
Umbenennen, 88, 121, 178
umorganisieren, 178
Umplatzen, 130
UNC Pfade, 16
unversionierte 'Arbeitskopie', 128
URL geändert, 130

V

vergleichen, 62, 77, 117
Vergleichen, 77
Vergleichsprogramme, 82
Verknüpfung, 182
verschieben, 88

Verschieben, 178
Version, 184
Versionierung entfernen, 182
Versionskontrolle, 1
Versionsnummer in Dateien, 167
Versionsprüfung, 184
Verweis, 19
verzweigen, 83, 101
ViewVC, 135
Vom Projektarchiv abkoppeln, 182
VS2003, 185

W

Web Sicht, 135
WebDAV, 26
Webseite, 19
WebSVN, 135
Windows Domäne, 31
Windows shell, 1
Wörterbuch, 3

Z

Zugriff, 16