



Subversion in der Web-Entwicklung

Subversion

Unterstützt durch CollabNet

www.collab.net

Aktuelle Version: 1.2.0

Unterstützte OS:

Linux, Win, Mac OSX, u.a.

Lizenztyp:

Subversion (Apache Style)

Website:

subversion.tigris.org

Quellen:

www.wikipedia.de

subversion.tigris.org

svn-book

Was ist “Versionsverwaltung”

Allgemein

Unter einer Versionsverwaltung versteht man ein System, welches typischerweise in der Softwareentwicklung zur Versionisierung und für den kontrollierten, gemeinsamen Zugriff auf Quelltexte, eingesetzt wird. Hierzu werden alle laufenden Änderungen erfasst und alle Versionsstände der Dateien in einem Archiv mit Zeitstempel und Benutzererkennung gesichert.

Es wird sichergestellt, dass jeder Benutzer mit dem aktuellsten Stand arbeitet oder auf Wunsch auf die archivierten Stände zugreifen kann. Dadurch ist eine Versionsverwaltung nicht nur für professionelle Entwickler in großen Teams, sondern auch für allein entwickelnde Entwickler interessant. Es kann jederzeit eine ältere Version aufgerufen werden, falls eine Änderung nicht funktioniert und man sich nicht mehr sicher ist, was nun alles geändert wurde.

Für Versionsverwaltungssysteme sind die Abkürzungen VCS (Version Control System) oder SCM (Source Code/Control Managementsystem) gebräuchlich.

Aufbau und Architektur

Das zentrale Archiv wird als Repository (engl. Behälter, Aufbewahrungsort) bezeichnet. Die meisten Systeme verwenden hierfür ein eigenes Dateiformat (oder eine Datenbank). Die Versionsverwaltungssoftware speichert dabei üblicherweise nur die Unterschiede zwischen zwei Versionen um Speicherplatz zu sparen. Dadurch kann eine große Zahl von Versionen archiviert werden. Durch dieses Speicherformat kann jedoch nur mit der Software des Versionsverwaltungssystems auf die Daten zugegriffen werden, die sodann die gewünschte Version bei einem Abruf unmittelbar aus den abgelegten Schnipseln „zusammenbaut“. Solche Software ist häufig als Client-Server-System aufgebaut, sodass der Zugriff auf ein Repository auch über Netzwerk erfolgen kann.

Damit die in der Softwareentwicklung eingesetzten Programme wie z.B. Compiler mit den im Repository abgelegten Dateien arbeiten können ist es erforderlich, dass jeder Entwickler sich den aktuellen (oder einen älteren Stand) des Projektes in Form eines Verzeichnisbaumes aus herkömmlichen Dateien erzeugen kann. Ein solcher Verzeichnisbaum wird als Arbeitskopie bezeichnet. Ein wichtiger Teil des Versionsverwaltungssystems ist ein Programm, das in der Lage ist, diese Arbeitskopie mit den Daten des Repositories zu synchronisieren. Das Übertragen einer Version aus dem Repository in die Arbeitskopie wird als Checkout oder Aktualisieren bezeichnet, während die umgekehrte Übertragung als Checkin oder Commit genannt wird. Solche Programme sind entweder kommandozeilenorientiert, mit grafischer Benutzeroberfläche oder als Plugin für integrierte Softwareentwicklungsumgebungen ausgeführt. Häufig werden mehrere dieser verschiedenen Zugriffsmöglichkeiten wahlweise bereitgestellt.

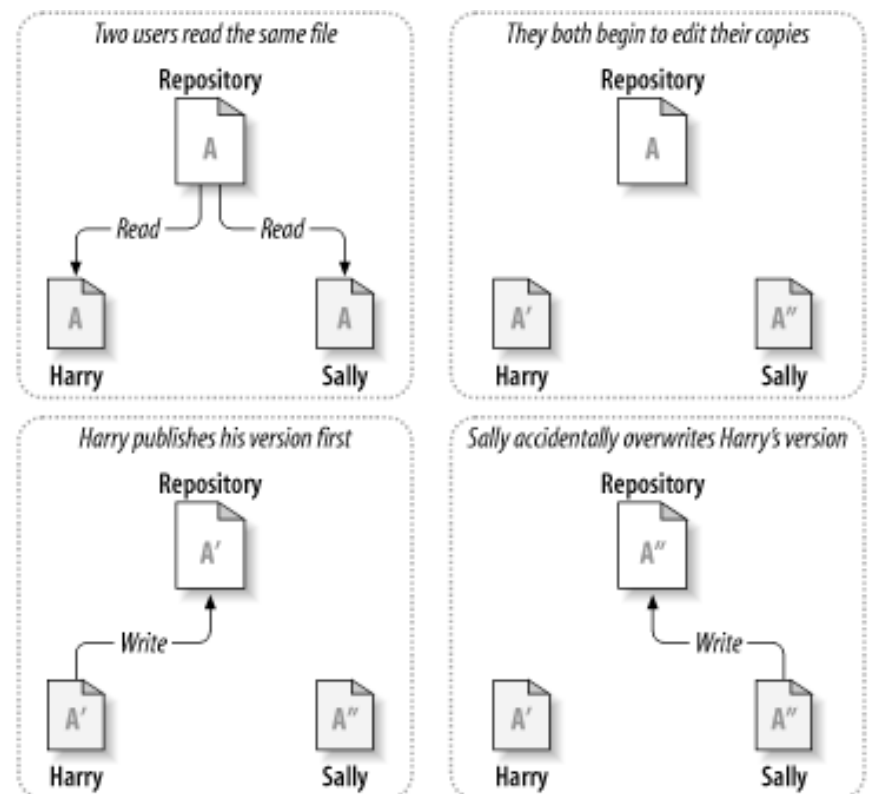
Grundsätzliche Arten der Versionierung

Die Hauptaufgabe eines Versionsverwaltungssystems ist es Daten für die gemeinsame Nutzung und Bearbeitung zur Verfügung zu stellen und zu verwalten.

Das Problem der „gemeinsamen Nutzung“

Alle Versionsverwaltungssysteme haben ein gemeinsames Problem zu lösen: wie ermöglichen Sie es Benutzern Daten gemeinsam zu bearbeiten, ohne dass der eine die Arbeit des anderen unbeabsichtigt überschreibt?

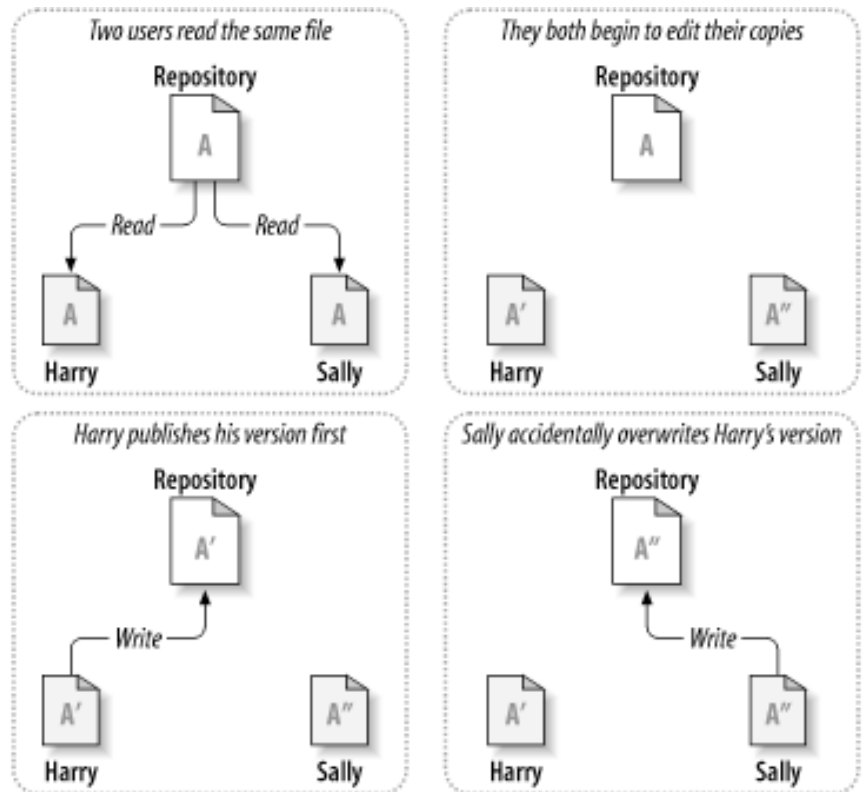
Man stelle sich folgendes Szenario vor. Harry und Sally lesen eine Datei aus einem für beide zugänglichen Repository. Beide bearbeiten die Datei. Harry ist schneller und führt seine Änderungen zurück in das Repository. Sally wird Zweite und auch sie schreibt Ihre Datei zurück. Damit hat sie die Arbeit von Harry zumindest im Repository vernichtet.



Um dieses Szenario zu vermeiden gibt es grundsätzlich zwei verschiedene Vorgehensweisen für ein SCM.

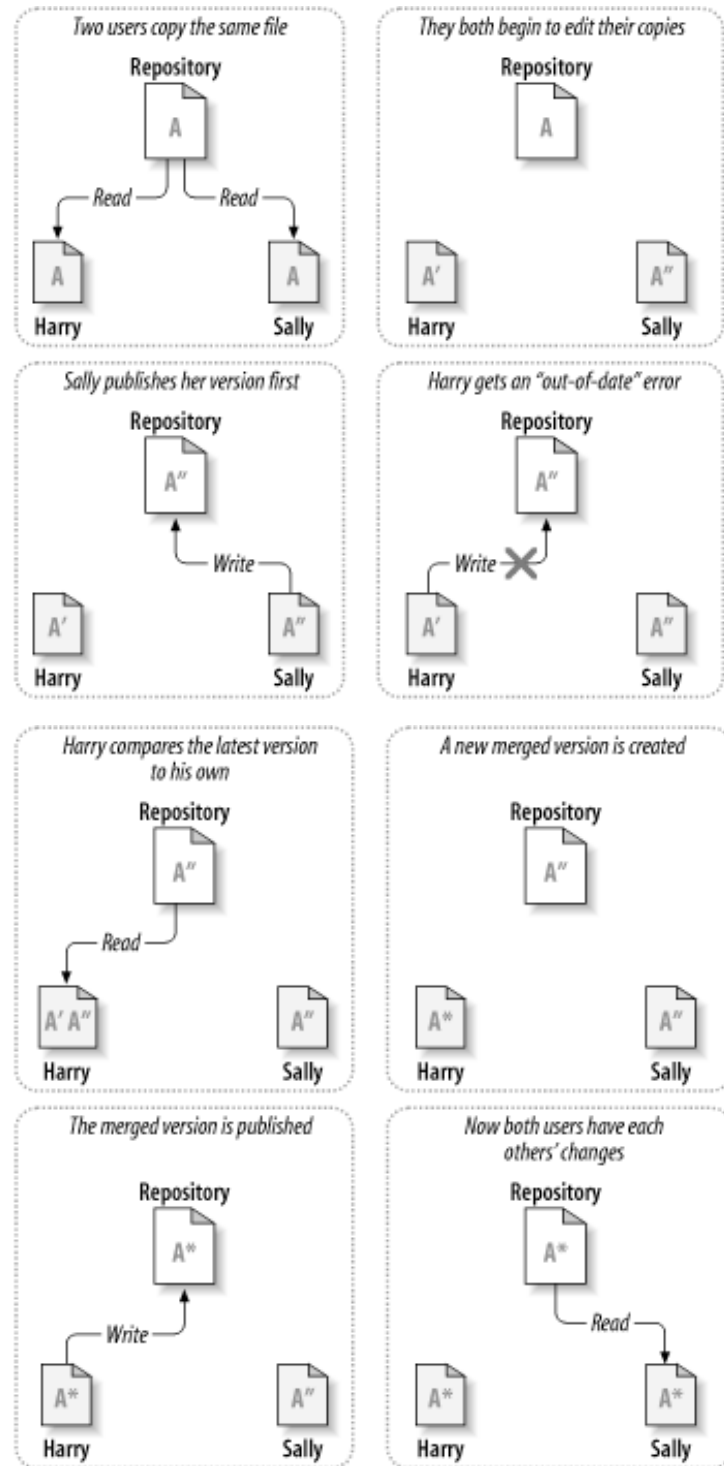
Lock-Modify-Unlock

Dies ist die traditionelle Arbeitsweise eines Versionsverwaltungssystems. Einzelne Dateien müssen vor einer Änderung durch den Benutzer gesperrt und nach Abschluss der Änderung wieder freigegeben werden. Während sie gesperrt sind, verhindert das System Änderungen anderer Benutzer.



Copy-Modify-Merge

Ein solches System lässt gleichzeitige Änderungen mehrerer Benutzer an einer Datei zu. Anschließend werden diese Änderungen automatisch oder manuell zusammengeführt (Merge). Somit wird die Arbeit des Entwicklers wesentlich erleichtert, da Änderungen nicht im Voraus angekündigt werden müssen. Insbesondere, wenn mehrere Entwickler räumlich getrennt arbeiten, wie es beispielsweise bei Open-Source Projekten der Fall ist, ermöglicht dies deutlich flüssigeres Arbeiten.



Links

SCM-Übersicht

<http://better-scm.berlios.de/comparison/comparison.html>

Bekannte Versionsverwaltungssysteme

Alienbrain

BitKeeper

Rational ClearCase

AccuRev

CMVC

CVS

Darcs

Git

GNU arch

in-Step

Monotone

Perforce

RCS

SCCS

Subversion

Visual SourceSafe (Microsoft)

Unter <http://better-scm.berlios.de/comparison/comparison.html> ist eine sehr umfangreiche Übersicht über 13 verschiedene Versionsverwaltungssysteme zu finden.

Links

Subversion Homepage

<http://subversion.tigris.org>

Subversion Handuch (englisch)

<http://svnbook.red-bean.com/>

Subversion

Subversion (SVN) ist eine Open-Source-Software zur Versionsverwaltung. Da es viele Schwächen des in Entwicklerkreisen sehr beliebten Programms CVS behebt, wird Subversion oft als dessen Nachfolger bezeichnet, obwohl es sich um ein eigenständiges Projekt handelt. Es ist jedoch absichtlich von der Bedienung sehr ähnlich gehalten. CVS-Umsteiger werden es deshalb zu schätzen wissen, dass man bei den meisten Befehlen lediglich das cvs durch svn ersetzen muss. Zusätzlich zu vielen neuen Features werden fast alle Funktionen von CVS unterstützt. Mit cvs2svn existiert ein Konverter, mit dem ein CVS-Repository zu Subversion konvertiert werden kann. Subversion wurde unter einer Apache Style Lizenz veröffentlicht.

Geschichte

Subversion wird seit Anfang 2000 bei CollabNet entwickelt und erreichte am 23. Februar 2004 die stabile Version 1.0. Am 29. September 2004 erschien Version 1.1, dessen größte Neuerung war, dass Repositories nicht mehr nur in einer Berkeley-Datenbank verwaltet werden können, sondern dass dazu auch direkt das Dateisystem benutzt werden kann. Außerdem wurden internationalisierte Programmausgaben ermöglicht. Die am 23. Mai 2005 erschienene Version 1.2 unterstützt nun auch optionale Bearbeitungssperren für Dateien, was teilweise für binäre Dateien von Vorteil sein kann.

Unterschiede zu CVS

Das Versionsschema von Subversion bezieht sich nicht mehr auf einzelne Dateien, sondern auf das ganze Repository. Somit kann man einfacher eine exakte Version beschreiben (z. B. „Version 2841“ statt „Version vom 23. März 2004 20:56:31 GMT“). Die Revisionsnummer einer Datei entspricht dabei der Revisionsnummer des Repositorys, als sie das letzte mal geändert wurde, die Revisionsnummer eines Verzeichnisses entspricht der höchsten Revisionsnummer der enthaltenen Dateien und Verzeichnisse. Die Revisionsnummern einer einzelnen Datei können also durchaus lückenhaft sein, wenn diese nicht bei jedem Commit geändert wurden. Beispielsweise könnte eine Datei bei der Revision 23 zum Repository hinzugefügt worden sein, und einmal in der Revision 48 und 52 verändert worden sein. Bei einem Checkout einer Datei wird die größte Revisionsnummer ausgecheckt, die kleiner oder gleich der angeforderten ist. Wird in dem Beispiel die Revision 52 angefordert, so wird die Revision 52 der Datei ausgecheckt, wird hingegen die Revision 51 angefordert, liefert Subversion die Revision 48.

Subversion speichert beim Checkout, Update und Commit in einem gesonderten Verzeichnis (.svn) eine zweite Kopie jeder Datei. Dadurch verdoppelt sich der Speicherbedarf einer Arbeitskopie ungefähr, allerdings bietet dies bei entfernten Repositorys auch einige Vorteile. So können einige Aktionen wie Anzeige der lokalen Änderungen komplett ohne Netzwerkzugriff erfolgen und Subversion kann auch beim Commit nur die geänderten Teile übertragen, während CVS bei der Übertragung zum Server jeweils die gesamte Datei übertragen muss.

Vorteile gegenüber CVS

In Subversion können Dateien (und Verzeichnisse) auch umbenannt und verschoben werden. In CVS muss eine Datei an der alten Stelle auf „Gelöscht“ gesetzt und an der neuen Stelle neu eingefügt werden, wobei die Historie der Datei nicht übernommen wird.

Subversion kann auch Verzeichnisse und Metadaten wie Dateiberechtigungen verwalten. Insbesondere können Verzeichnisse auch als gelöscht markiert werden. Dies ist mit CVS nicht möglich, hier können nur optional leere Verzeichnisse beim Checkout ignoriert werden.

Subversion bietet einen verbesserten Umgang mit Binärdateien. Es erkennt solche Dateien (beispielsweise Bilder oder Audiodateien) automatisch und es werden (wie bei Textdateien) nur die Differenzen zwischen den geänderten Versionen gespeichert. In CVS geht das umständlicher über Eintrag von binären Dateitypen (deren Endung) in cvs wrappers und wobei diese Filetypen voll gespeichert werden.

Commits sind in Subversion atomar, weshalb eine Änderung entweder komplett oder gar nicht ins Repository gespeichert wird. Verbindungsabbrüche und mehrere, gleichzeitige Commits können somit nicht zu inkonsistenten Zuständen führen.

Zusätzlich zu einem eigenen Server und der Speicherung im lokalen Dateisystem existiert auch ein Modul für den Apache 2 Webserver, mit dem die Daten auch mit der HTTP/HTTPS-Erweiterung WebDAV übertragen werden können.

Einige Operationen (update, tagging, branching) sind deutlich schneller als bei CVS.

Nachteile gegenüber CVS

Die gleichen Daten (z.B. nach Konvertierung) benötigen in einem Subversion-Repository deutlich mehr Platz als in einem CVS-Repository.

Subversion ist noch nicht so gut getestet wie CVS.

Benötigt den Apache HTTP Server 2.0, wenn das WebDAV-Feature genutzt werden soll. Allerdings könnte man hier entgegen, dass CVS WebDAV gar nicht unterstützt. Außerdem gibt es auch ein eigenes Protokoll, welches vergleichbar mit dem Netzwerkprotokoll von CVS ist.

Systemaufbau

Komponenten

Subversion basiert auf einigen Bibliotheken, von denen jedoch nur die Apache Portable Runtime zwingend erforderlich ist.

Wenn das Repository in einer Berkeley-Datenbank gespeichert werden soll, wird diese in einer Version ≥ 4.0 benötigt.

XML-Parser Expat

Python 2.x

Apache 2

OpenSSL oder andere SSL-Implementation

Neon

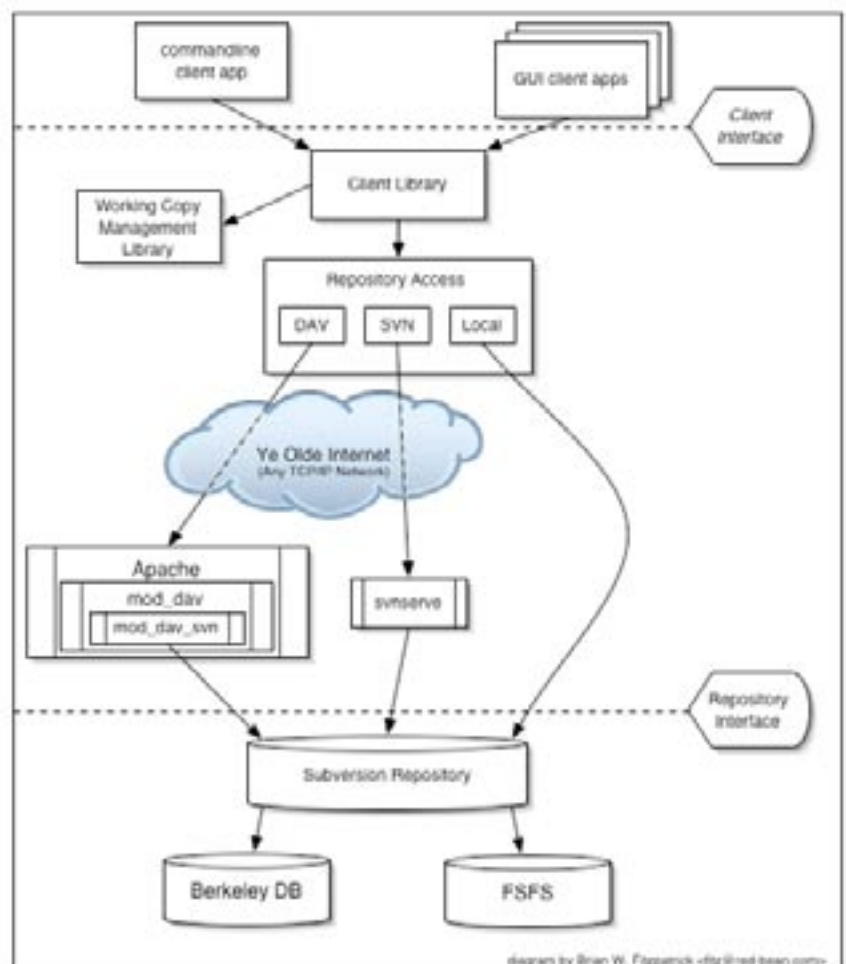
Architektur

Die Basis bildet das Repository, ein Dateisystem, das entweder in einer Datenbank oder im Filesystem des Rechners abgebildet wird. Mithilfe des Subversion Clients kann auf dieses Repository zugegriffen werden. Der Client verwaltet die lokale Kopie des Repositories als sog. working copy.

Der Zugriff auf das Repository wird durch die RA-Ebene (Repository Access) gesteuert. Es gibt eine ganze Reihe verschiedener Zugriffsmöglichkeiten:

Lokaler Zugriff (`file:///home/ralf/dev/projekt/...`)

Netzzugriff über `svn://...`, `svn+ssh://...` und `http://...` bzw. `https://...`



Repository Filesystem

Seit der Version 1.1 kann subversion das Repository auch im Dateisystem des Rechners verwalten (FSFS). Die bisherige einzige Alternative das Repository in einer BerkeleyDB zu verwalten stieß insbesondere an Performance-Grenzen bei umfangreichen Schreiboperationen und grossen Verzeichniss-Rekursionen.

Wenn man subversion aus den Quellen kompiliert und eine passende BerkeleyDB-Installation nicht gefunden wurde, wird ersatzweise auf FSFS zurückgegriffen.

API und Language-Bindings

Subversion bietet eine umfangreiche API und eine ganze Reihe von Implementierungen für die gängigsten Programmiersprachen. Subversion selbst ist in C geschrieben und stellt die Core- und Client APIs zur Verfügung.

Implementierungen dieser APIs sind u. a. in Java, C, C++, Python, und Perl verfügbar. Daneben werden auch andere Sprachen unterstützt, die den SWIG-Mechanismus unterstützen.

Unterstützung in PHP ist zum einen in Form des subversion Client Wrappers `pera::svn` gegeben, der in PEAR gehostet wird. Zum anderen haben sich aber auch die „harten“ Jungs von der PECL-Fraktion daran gemacht, eine PHP-extension zu implementieren, die die Client-Funktionalitäten abdeckt.

Anwendungsstruktur eines Repositories

Um seine Projekte überschaubar und nachvollziehbar mittels eines Versionsverwaltungssystems zu verwalten, sollte man sich einige Gedanken machen über die Struktur einer solchen Ablage.

Unter Umständen finden sich in einem Repository auch Informationen wieder, die nur mittelbar mit dem Projekt zu tun haben, wie etwa Whitepapers, Konzepte, Illustrationen etc..

Neben der Ablage der Daten ist auch zu beachten, dass bei einem Projekt u.U. benannte Versionen (Meilensteine, spezielle Zustände) und vielleicht sogar ganze Entwicklungszweige, also unabhängige Stränge mit eigenem Entwicklungsverlauf ergeben.

Die Entwickler von Subversion schlagen folgendes grundsätzliches Layout vor:

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

Im trunk-Verzeichnis wird das eigentliche Projekt geführt. In branches und tags werden die Entwicklungszweige bzw. benannten Versionen abgelegt.

Typischer Workflow

Beim Aufsetzen eines neuen Projektes unter Kontrolle eines Versionsverwaltungssystems sind die folgenden, notwendigen Schritte durchzuführen. Diese Schritte sind auch für die Überführung eines bestehenden Projektes anzuwenden.

Notwendige Arbeitsschritte

- Erzeugen eines Repositories

- Erster Import der vorhandenen Projektdaten in einer Grundstruktur

- Anlegen einer Arbeitskopie (working copy) mittels checkout

Die Arbeit mit subversion setzt die existenz eines Repositories voraus. Daher muss zunächst ein solches Repository erzeugt werden:

```
$ svnadmin create /path/to/repos
```

Dieses Verzeichnis /path/to/repos enthält danach die für subversion notwendigen Verzeichnisse, initialisierten Datenbanken, Indexdateien etc.

```
$ ls /path/to/repos
conf/ dav/ db/ format hooks/ locks/ README.txt
```

Als nächstes sollten sie die oben beschriebene Struktur anlegen und mit evtl. schon vorhandenen Dateien füllen

```
/tmp/project/branches/
/tmp/project/tags/
/tmp/project/trunk/
                                foo.php
                                bar.php
                                images/php.png
                                README.txt
```

Diese Struktur bzw. dieser Projektstand kann nun in das Repository überführt werden (import).

```
$ svn import /tmp/project file:///path/to/repos \
-m „initial import“
Adding /tmp/project/branches
Adding /tmp/project/tags
Adding /tmp/project/trunk
Adding /tmp/project/trunk/foo.php
Adding /tmp/project/trunk/bar.php
Adding /tmp/project/trunk/images/php.png
Adding /tmp/project/trunk/README.txt
...
Committed revision 1.
$
```

Nun liegen die unter /tmp/project liegenden Dateien und Verzeichnisse als verwaltete Version 1 im Repository vor.

Um jetzt am Projekt weiter arbeiten zu können, ist es notwendig, eine Arbeitskopie (working copy) anzulegen. Da diese Arbeitskopie nicht nur die Dateien und Verzeichnisse unseres ursprünglichen Imports enthalten wird, sondern auch subversion spezifische Dateien, ist es notwendig, diese aus dem Repository zu entnehmen.

```
$ svn checkout file:///path/to/repos/trunk project
A project/foo.php
A project/trunk/bar.php
A project/trunk/images/php.png
A project/trunk/README.txt
...
Checked out revision 1.
```

Jetzt haben wir eine persönliche Arbeitskopie unseres Projektes in einem neuen Unterverzeichnis project und können mit der Arbeit beginnen.

Wechseln Sie in das Working Copy-Verzeichnis

Nehmen Sie Änderungen in den Dateien vor

Benutzen Sie svn diff um die Änderungen im diff-Format zu sehen

Benutzen sie svn stat um eine Übersicht der geänderten Dateien zu erhalten

Schreiben Sie die Änderungen zurück in das Repository

Im svn-book ist ein umfangreiches Tutorial als Guided Tour zu finden.

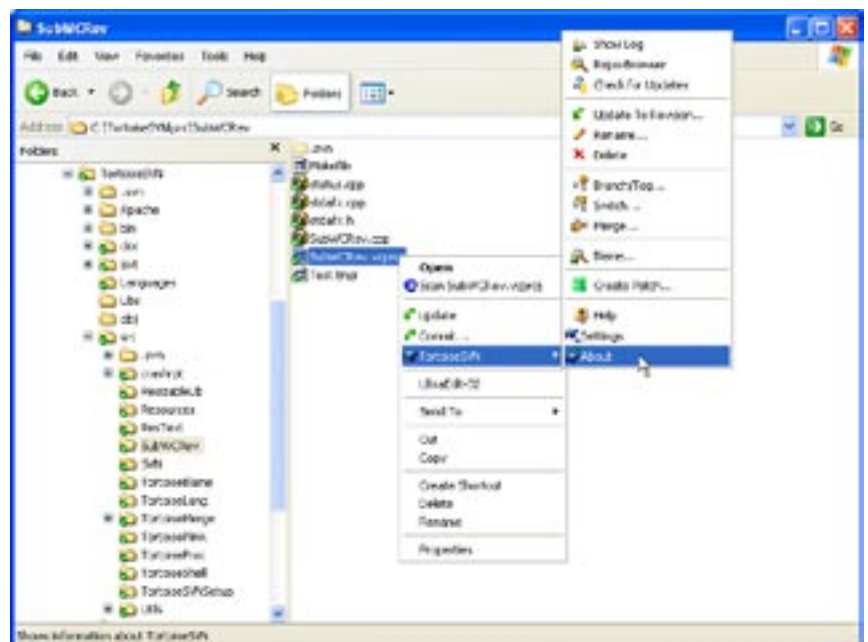
Clients, GUI, Repository Browsers

Neben dem subversion commandline client gibt es eine ganze Reihe von Clients die auf der subversion Client API aufbauen bzw. einen wrapper um den CLI-client nutzen.

Es gibt inzwischen für eine ganze Reihe von Integrierten Entwicklungsumgebungen (IDE) eine subversion-Anbindung. Die PHP-IDEs Maguma workbench und ZEND haben eine solche zumindest angekündigt.

Eine Standardschnittstelle für die Integration von Werkzeugen für die Versionsverwaltung in IDEs ist Microsoft SCC API. Für Subversion gibt es z.Zt. zwei SCC-Provider, tortoisessvncc und subway.

Auch die Integration in den Desktop bzw. Explorer ist erfolgt. Mit TortoiseSVN steht der Windows-Explorer mit all seinen Stärken und Schwächen als grafischer Client für die Arbeit in der Arbeitskopie und zur Kommunikation mit dem Repository zur Verfügung. Ähnliche Projekte gibt es auch für den Linux-KDE-Desktop und Mac OS X-Finder.



Mit dem PHP-basiertem webSVN und python-basiertem viewCVS stehen zwei leistungsfähige web-gestützte Repository-Browser bereit, die neben grafischen Diffs auch Syntax-Highlighting und Multimedia-Dateiformate unterstützen.

Tools und mehr

Darüberhinaus gibt es eine ganze Hand voll Konvertern, die die Überführung von Projekten aus verschiedenen SCM-System nach subversion ermöglichen.

SCM-Bug ist eine Middleware, die zwischen SCM-Systemen (cvs und subversion) und Bug-Tracking-Systemen wie Bugzilla und Mantis vermittelt und issues bzw. tickets Versionen zuordnen kann.

Neben einer wachsenden Liste von subversion-Providern gibt es auch eine ganze Reihe von Projekten die subversion integrieren, z.B. gforge oder trac als Projekt-Plattformen für Entwicklergruppen.