



Subversion

Ein besseres CVS?

Steffen Schluff

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Version: 1.0

Gliederung

- Subversion Grundlagen
- Dateien unter Subversion Kontrolle stellen
- Mit Dateien in Subversion arbeiten
- Markierungen und Zweige in Subversion
- Properties
- Zusammenfassung

2

Gliederung



- **Subversion Grundlagen**
- Dateien unter Subversion Kontrolle stellen
- Mit Dateien in Subversion arbeiten
- Markierungen und Zweige in Subversion
- Properties
- Zusammenfassung

3

Was ist Subversion ?



- Werkzeug zur Versionsverwaltung
 - Open Source Projekt gefördert durch CollabNet
- Nachfolger von CVS (Concurrent Versions System)
 - „[...] goal [...] is a compelling replacement for CVS [...]“
 - Schwächen von CVS beseitigen bei Beibehaltung der Grundidee
- Historie
 - Frühjahr 2000 - Entwicklungsbeginn
 - Sommer 2001 - Subversion „self-hosting“
 - Frühjahr 2004 - Version 1.0.0
 - April 2005 - Versionen 1.1.4 bzw. 1.2.0-rc1

4

Subversion Verfügbarkeit



- Verfügbar auf allen gängigen Betriebssystemen
 - Portabilität durch ANSI C und APR (Apache Portable Runtime) Library
- Downloads beinhalten alle notwendigen Kommandozeilentools
 - Client, Server, Apache Server Module und diverse Tools
- Diverse graphische Clients und Tools verfügbar
 - TortoiseSVN, Subclipse Plugin, ViewCVS, SVN::Mirror
- Standardreferenz „Version Control with Subversion“
 - Als freies Buch online verfügbar
- Language Bindings für Subversion APIs
 - Bindings für diverse Sprachen (Perl, Python, Java) verfügbar

5

Subversion Repository



- Zentraler Punkt ist ein Repository
 - Serverseitiger Datenspeicher
 - Wird erzeugt über svnadmin Tool
- Speichert Information in Form eines „virtuellen“ Dateisystems
 - Clients können Dateisystem lesen und schreiben
- Jeder Schreibzugriff wird historisiert
 - „Fileserver mit Historie“
- Zwei Implementierungen des Subversion Dateisystems
 - Berkeley DB
 - **Initiale Implementierung**
 - FSFS (Fairly Secure File System)
 - **Neu ab Version 1.1, Default ab Version 1.2**

6

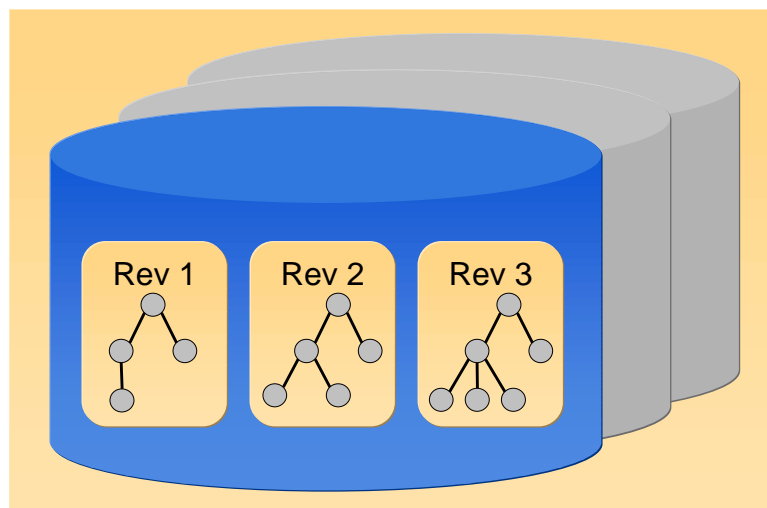
Subversion Revisions



- Jeder Schreibzugriff erzeugt neue Revisionen
 - Serverseitiger Versionszähler des Dateisystems
- Revisionen beziehen sich auf gesamten (!) Dateisystembaum
 - Größter Unterschied zu CVS (Datei bezogene Revisionen)
- Eine einzelne Datei muß sich nicht in jeder Revision ändern
 - Gegenteil ist wahrscheinlicher
 - Datei „Foo“ in R5 kann identisch sein zu Datei „Foo“ in R6
 - Nicht „Revision 5 von ‚Foo‘“ sondern ‚Foo‘ in Revision 5“
- Repository verwaltet ein Array von Dateisystemen
 - Revisionsnummer ist Index für dieses Array

7

Subversion Server mit mehreren Repositories



8

Working Copies



- Client erhält Arbeitskopie von Subversion Server
 - Lokaler Verzeichnisbaum des Clients
 - Entspricht Teilbaum des Subversion Dateisystems
 - Wird durch Benutzer mit Subversion Dateisystem abgeglichen
- Subversion verwendet „Copy-Modify-Merge“ Modell
 - Benutzer arbeiten stets parallel ohne Sperren
 - Überschneidende Änderungen durch Benutzer zu synchronisieren
 - Gegenteil „Lock-Modify-Unlock“ Model
- Sperren von Dateien (File Locking) ab Subversion 1.2
 - Konkurrierende Änderungen sind in Binärformaten schwierig
 - Erzwingen von serialisiertem Zugriff auf bestimmte (Binär-) Dateien

9

Server Konfiguration



- Angabe der Repositoryadresse durch URL (svn://svn.oio.de/dev)
 - file:/// - Zugriffsrechte des Benutzers auf Repository Dateien
 - http:/// - Anpassen der Apache Konfigurationsdatei „httpd.conf“
 - svn:// - Einrichten (x)inetd und Anpassen „svnserve.conf“ Datei
- Weitere Anpassungen über Repository spezifische Hooksripts
 - Bestimmte Aktionen lösen serverseitige Skripte aus
- Pflege und Wartung eines Repository über Administrationstools
 - Zentrale Tools svnlook und svnadmin (auch Backup und Recovery)
 - Viele Zusatzskripte in Subversion Quellcode verfügbar
- Eigenes Tool für Konvertierung vorhandener CVS Repos
 - Python basiertes Tool cvs2svn für einmalige (!) Konvertierung

10

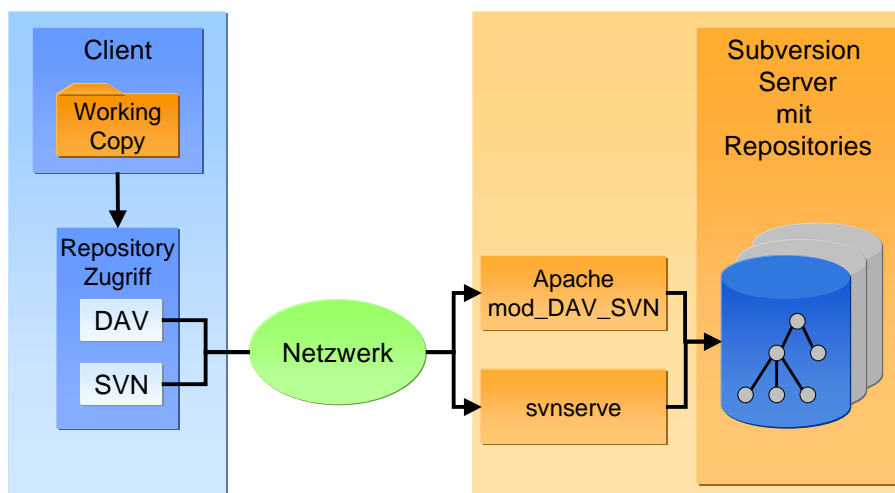
Client Konfiguration



- Clientseitige Konfiguration in „Runtime Configuration Area“
- Werte werden im Standard INI - Format abgelegt
 - Trennung systemweite und userspezifische Bereiche
- Enthält folgende Dateien
 - readme.txt - Dokumentation zu INI - Format
 - servers - Einstellungen zur Netzwerkschicht, z.B. Proxies
 - config - Sonstige Einstellungen des Clients, z.B. Properties
- „Runtime Configuration Area“ enthält auch „auth“ Verzeichnis
 - Cache für Authentifizierung gegenüber Server
 - „Pull“ der Credentials im Unterschied zu CVS „Push“ Ansatz

11

Subversion aus der Vogelpersicht



12

Gliederung



- Subversion Grundlagen
- **Dateien unter Subversion Kontrolle stellen**
- Mit Dateien in Subversion arbeiten
- Markierungen und Zweige in Subversion
- Properties
- Zusammenfassung

13

Initiale Projektstruktur



- Branches und Tags haben keine Sonderstellung in Subversion
- Abbildung über „normale“ Repository Verzeichnisse
- Initiale Projektstruktur sollte dies bereits berücksichtigen

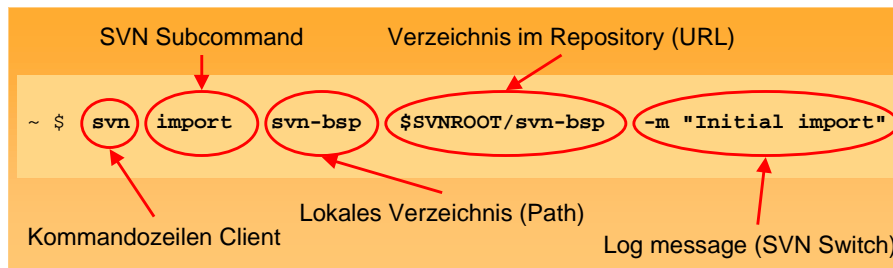


14

Aufbau eines Subversion Befehls



- Keine unterschiedlichen (Kommando-) Optionen wie bei CVS
- Subversion verwendet Switches
 - Switches haben immer die gleiche Bedeutung
 - Nicht jedes Subcommand verwendet alle Switches



15

Import kontrollieren



- Repository Inhalt ohne (kompletten) Download zu untersuchen
- Subversion Subcommands `list` und `cat`
- Subversion sieht Netzwerk Bandbreite als limitierte Ressource

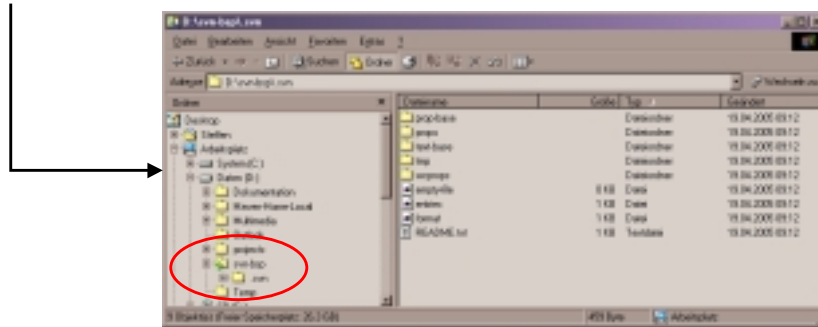
```
~ $svn list $SVNROOT/svn-bsp  
  
branches/  
tags/  
trunk/
```

16

Initiales Auschecken

Repository URL (Projekt) Lokales Verzeichnis (Path)

```
~ $svn checkout $SVNROOT/svn-bsp/trunk svn-bsp
```



17

Clientseitiger Zustand

- Verwaltungsverzeichnisse names „svn“ auf Clientseite
 - Vergleichbar mit CVS Verzeichnissen
- Verzeichnis enthält unveränderte Kopien aller Arbeitsdateien
 - Letzte Original Version aus Repository
 - Weitere Informationen wie Repository-URL oder Properties
- Erlaubt viele Aktionen ohne Server-Interaktion (offline) auszuführen
 - Vergleiche mit ursprünglicher Repository Version
 - Lokale Änderungen prüfen, anzeigen, rückgängig machen
- Geringere Netzlast beim Einchecken von Änderungen
 - Lokale Änderungen können vor dem Einchecken berechnet werden

18

Neue Datei in Subversion ablegen



```
~/svn-bsp $ls
Hello.java

~/svn-bsp $svn add Hello.java
A      Hello.java

~/svn-bsp $svn commit -m "Initial Import" Hello.java
Hinzufügen      Hello.java
Übertrage Daten .
Revision 6 übertragen.
```

19

Binäre Dateien unter Subversion



- Mögliche Probleme mit Binärdateien
 - Probleme beim CR / LF - Konvertierung
 - Probleme mit Keywords
 - Keine automatische Diff-Unterstützung
- Subversion erkennt Binärdateien automatisch
 - Kein spezielles Markieren über `svn add -kb`
 - Subversion Server Algorithmen (vdelta, xdelta) auch für Binärdateien
- Kein automatisches zeilenbasiertes Merging
 - Stattdessen Ablage der jeweiligen Originaldatei
 - Sperren von Dateien (File Locking) ab Subversion 1.2

20

Änderungen an der lokalen Arbeitskopie



- Zwei mögliche Arten von Änderungen
 - Dateibezogene und strukturelle Änderungen
- Datei bezogene Änderungen
 - Von Subversion verwaltete Datei editieren
 - Änderungen mittels `commit` übergeben
- Strukturelle Änderungen
 - Dateien / Verzeichnisse hinzufügen, löschen, kopieren, verschieben
 - Strukturelle Änderung durch `commit` bestätigen
 - Manche Änderungen auch direkt im Repository möglich, z.B. kopieren
 - (Bekannteste) Erweiterung gegenüber CVS

21

Verzeichnisse erzeugen und Dateien bewegen



```
~/svn-bsp $svn mkdir src
A      src

~/svn-bsp $svn commit -m "Created source directory." src
Hinzufügen      src

Revision 11 übertragen.

~/svn-bsp $svn move Hello.java src/
A      src\Hello.java
D      Hello.java

~/svn-bsp $svn commit -m "Moved file."
Löschen      Hello.java
Hinzufügen      src\Hello.java

Revision 12 übertragen.
```

22

Bewegte Dateien behalten Historie



- Subcommand log ist praktischer als CVS Pendant
 - Logging Ausgaben beziehen sich auf einzelne (atomare) Commit
 - Revision Keywords als symbolische Revisionsbezeichner

```
~/svn-bsp/src $svn -v log Hello.java -r BASE:6
-----
r12 | sschluff | 2005-04-19 12:05:17 +0200 (Di, 19 Apr 2005) | 1 line
Geänderte Pfade:
  D /svn-bsp/trunk/Hello.java
  A /svn-bsp/trunk/src/Hello.java (von /svn-bsp/trunk/Hello.java:6)

Moved file.
-----
r6 | sschluff | 2005-04-19 10:22:17 +0200 (Di, 19 Apr 2005) | 1 line
Geänderte Pfade:
  A /svn-bsp/trunk/Hello.java

Initial Import
-----
```

23

Status einer Datei anzeigen



- CVS Variante von status war unpraktisch
 - Unleserlicher Output, Vermischung lokaler und Repository Änderungen
 - CVS update Befehl in Praxis bevorzugt
- Subversion besitzt überarbeitetes status Subcommand
 - Gut les- und parsebares Ausgabeformat
 - Per Default kein Verbindungsaufbau mit dem Subversion Repository

```
~/svn-bsp/src $svn status --verbose --show-updates
M      *      12      12 sschluff   Hello.java
        *      11      11 sschluff   .
Status bezogen auf Revision: 12
```

24

Gliederung



- Subversion Grundlagen
- Dateien unter Subversion Kontrolle stellen
- **Mit Dateien in Subversion arbeiten**
- Markierungen und Zweige in Subversion
- Properties
- Zusammenfassung

25

Änderungen übergeben oder zurücknehmen



- Lokale Änderungen per `commit` an Repository übergeben
 - Durch Original Datei in „svn“ Verzeichnis lokales Diff möglich
 - Nur Delta muß übertragen werden (geringere Netzwerklast)
- Lokale Änderungen per `revert` zurück nehmen
 - Neues Subversion Subcommand, keine Entsprechung in CVS
 - Erfordert keine Verbindung zum Server
 - In CVS lokales löschen und anschließend `update`

```
~/svn-bsp/src $svn revert Hello.java
Rückgängig gemacht: 'Hello.java'
```

26

Update ausführen

- Garantiert fehlerfrei, wenn keine lokale Änderungen

```
~/svn-bsp-die-zweite $svn update
U src\Hello.java
Aktualisiert zu Revision 13.
```

27

„Wer zu spät kommt...“

```
public class Hello {
    public static void main (String[] args) {
        System.out.println("----+++");
        System.out.println("Hello World");
        System.out.println("----+++");
    }
}
```

Client 1

```
public class Hello {
    public static void main (String[] args) {
        System.out.println("----*****");
        System.out.println("Hello World");
        System.out.println("----*****");
    }
}
```

Client 2

1

2



28

Commit nicht möglich



- Subversion prüft, ob zwischenzeitlich neuere Revision verfügbar
 - Kein Commit wenn nicht relativ zur zuletzt geänderte Dateirevision

```
~/svn-bsp-die-zweite/src $svn -vu status
M *      13      13 sschluff  Hello.java
          13      13 sschluff  .
Status bezogen auf Revision:    14

~/svn-bsp-die-zweite/src $svn commit \
                        -m "Output verbessert" Hello.java

Sende      Hello.java
svn: Übertragen fehlgeschlagen (Details folgen):
svn: Out of date: '/svn-bsp/trunk/src/Hello.java' in transaction 'i'
```

29

Update vor Commit notwendig



- Unvermeidbarer Update kann zu Konflikt führen
 - Alle notwendigen Dateien für Merge werden lokal bereit gestellt
 - Im Konfliktfall ist die Arbeitskopie bis zur Bereinigung „gesperrt“
 - Deutlich komfortabler als bei CVS

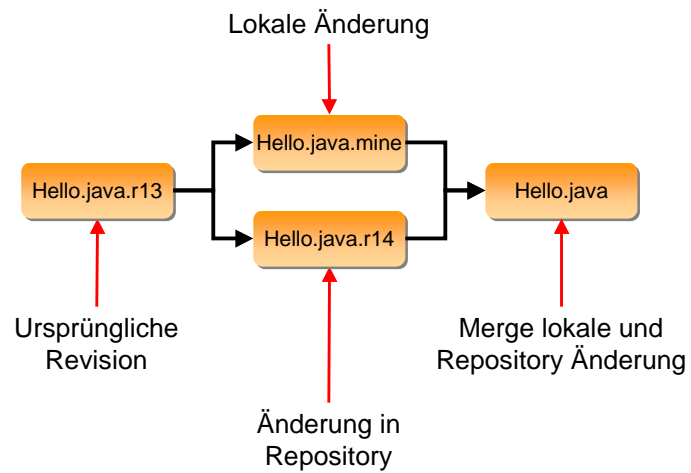
```
~/svn-bsp-die-zweite/src $svn update
C Hello.java
Aktualisiert zu Revision 14.

~/svn-bsp-die-zweite/src $ls
Hello.java Hello.java.mine Hello.java.r13 Hello.java.r14

~/svn-bsp-die-zweite/src $svn commit -m "Output verbessert"
Hello.java
svn: Übertragen fehlgeschlagen (Details folgen):
svn: Übertragung abgebrochen:
'c:/Steffen/svn-bsp-die-zweite/src/Hello.java' bleibt im Konflikt
```

30

Merge Information nach Update direkt vor Ort



31

Sperre nach Konflikt aufheben



- Nach Update entstandene Konflikte bereinigen
- „Konfliktsperre“ aufheben und bereinigte Revision übergeben
 - Neues Subcommando `resolved` in Subversion
 - Löscht auch die für Merge benötigten Dateien

```
~/svn-bsp-die-zweite/src $xemacs Hello.java
~/svn-bsp-die-zweite/src $svn resolved Hello.java
Konflikt von 'Hello.java' aufgelöst
~/svn-bsp-die-zweite/src $ls
Hello.java
~/svn-bsp-die-zweite/src $svn commit -m "Output verbessert" Hello.java
Sende      Hello.java
Übertrage Daten .
Revision 15 übertragen.
```

32

Konflikte automatisch lösen lassen



- Automatischer Merge bei „konfliktfreien“ Änderungen
 - Merge in der lokalen Arbeitskopie, da immer noch Fehlerpotential
 - Benutzer muß automatischen Merge durch commit bestätigen

```
~/svn-bsp-die-zweite/src $svn commit -m "Kommentar ergänzt" Hello.java
Sende      Hello.java
svn: Übertragen fehlgeschlagen (Details folgen):
svn: Out of date: '/svn-bsp/trunk/src/Hello.java' in transaction 's'

~/svn-bsp-die-zweite/src $svn update
G Hello.java
Aktualisiert zu Revision 21.

~/svn-bsp-die-zweite/src $svn commit -m "Kommentar ergänzt" Hello.java
Sende      Hello.java
Übertrage Daten .
Revision 22 übertragen.
```

33

Änderungen durch Merge zurücknehmen



- Berechnung eines „changeset“ um alte Version wiederherzustellen
 - Subcommand merge mit Revisionsangabe
 - Erzeugt lokale Änderung, die inhaltsgleich zur alten Version ist
 - Commit der inhaltsgleichen Version wird neue Revision

```
~/svn-bsp/src $svn merge -r 23:21 Hello.java
U Hello.java
~/svn-bsp/src $svn revert Hello.java
Rückgängig gemacht: 'Hello.java'
~/svn-bsp/src $svn merge -r BASE:PREV Hello.java
U Hello.java
~/svn-bsp/src $svn commit -m "Undoing change committed in r22" Hello.java
Sende      Hello.java
Übertrage Daten .
Revision 24 übertragen.
```

34

Gelöschte Dateien restaurieren



- Löszeitpunkt (Revision) durch `log` ermitteln
 - Dateien sind in alten Revisionen noch existent
- Datei durch Subcommand `copy` wieder in lokale Arbeitskopie legen
 - Subcommand `merge` ungeeignet da Seiteneffekte möglich

```
~/svn-bsp $svn log --verbose
r61 | schluffi | 2005-04-26 08:13:15 +0200 (Di, 26 Apr 2005) | 1 line
Geänderte Pfade:
  D /svn-bsp/trunk/foo.txt
Removed file
-----
...
~/svn-bsp $svn copy -r60 $SVNROOT/svn-bsp/trunk/foo.txt .
A
  foo.txt

~/svn-bsp $svn ci -m "resurrected file" foo.txt
Hinzufügen    foo.txt

Revision 62 übertragen.
```

35

Gliederung



- Subversion Grundlagen
- Dateien unter Subversion Kontrolle stellen
- Mit Dateien in Subversion arbeiten
- **Markierungen und Zweige in Subversion**
- Properties
- Zusammenfassung

36

Release erstellen



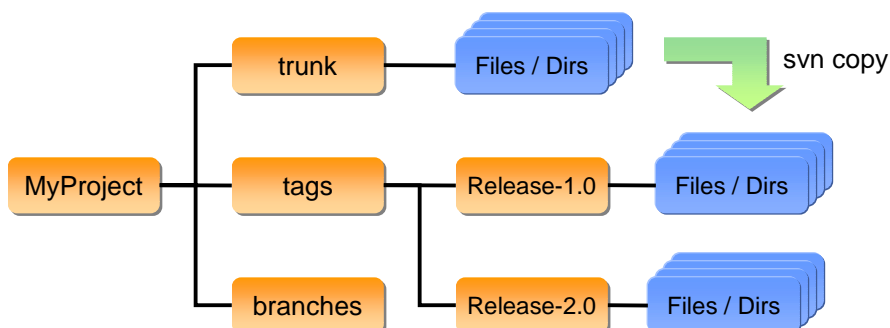
- Momentaufnahme eines Projekts
 - Kennzeichnet bestimmte Version
 - Entspricht dem CVS tag Befehl
- Jede Subversion Revision ist eine Momentaufnahme
 - Revision ist Momentaufnahme des gesamten Repository
 - Atomare Commits verhindern „ungültige“ Zwischenzustände
- Revisionsnummer ist nicht anwenderfreundlich
 - Besser „release-1.0“ für einen Teilbaum des Repository
- Lösung ist Kopie eines Repository Teilbaum zu erstellen
 - Bestimmtes Repository Verzeichnis wird als Tag betrachtet
 - Subversion Kopien sind „billig“ (vergleichbar „hard-link“)

37

Tag erstellen



- Branches und Tags haben keine Sonderstellung in Subversion
 - Unterschied zu CVS, keine Extra „Dimension“ für Branches und Tags
 - Bedeutung wird einem Repository Zweig durch User verliehen



38

Symbolische Markierung erstellen

- Zu taggende Version als lokale Arbeitskopie erstellen
 - Mixed Revisions je nach Bedarf über update Subcommand möglich
- Basierend auf Revisionen der Arbeitskopie Markierung erstellen
 - Durch Subcommand `copy` in entsprechendes Repository Verzeichnis

```
~/svn-bsp $svn copy . $SVNROOT/svn-bsp/tags/release-1.0 \  
-m "Tagging Rel 1.0"
```

Revision 28 übertragen.

```
~/svn-bsp $
```

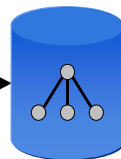
39

Fehler im Release entdeckt

```
public class Hello {  
    public static void main ( )  
    {  
        System.out.println("-----");  
        System.out.println("Hello World");  
        System.out.println("-----");  
    }  
  
    public String foo(){  
        if (true) {  
            throw new RuntimeException("BUG");  
        }  
        return "Fancy feature";  
    }  
}
```

Client 1

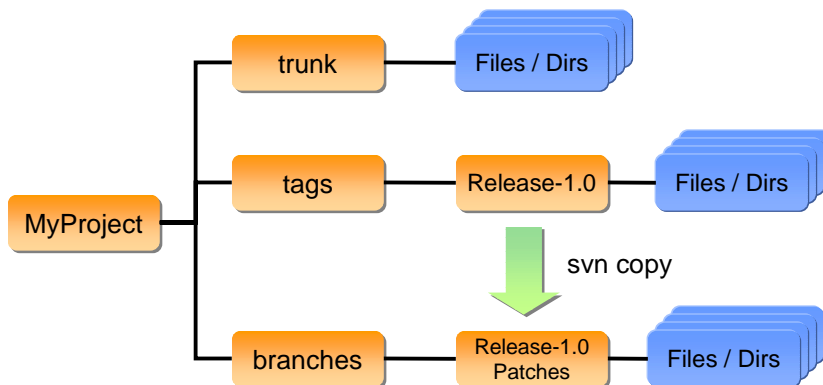
Neue Version
Foo 1.0



40

Branch erstellen

- Branches und Tags haben keine Sonderstellung in Subversion
 - Repository Verzeichnis das Release enthält für Branch kopieren



41

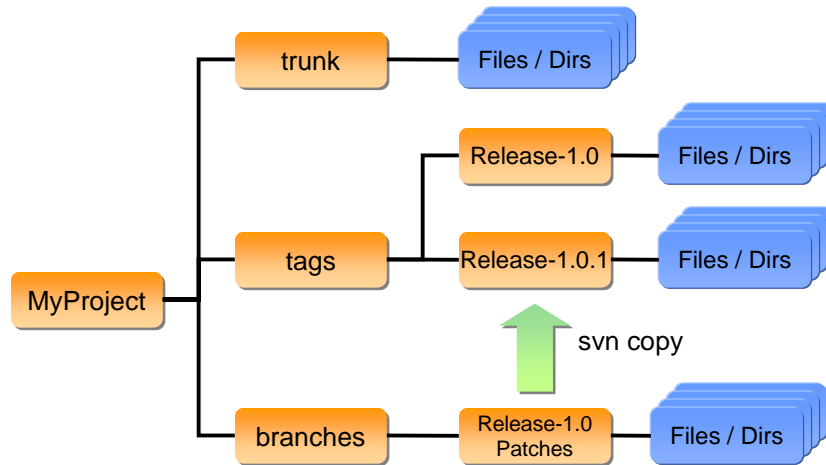
Branch erzeugen

- Nach Brancherzeugung Arbeitskopie auf Branch umstellen
 - Auschecken oder Subcommand `switch` auf bestehender Kopie
- Möglichkeit nur ein Verzeichnis auf Branch umzustellen
 - Gezielte Änderungen in Verzeichnis, Rest wird aus Trunk aktualisiert

```
~/svn-bsp $svn copy \  
$SVNROOT/svn-bsp/tags/release-1.0 \  
$SVNROOT/svn-bsp/branches/release-1.0-patches \  
-m "Creating a branch for Release 1.0"  
  
Revision 38 übertragen.  
  
~/svn-bsp $svn info | grep URL  
URL: file:///c:/programme/subversion/svnrepo/svn-bsp/trunk  
  
~/svn-bsp $svn switch $SVNROOT/svn-bsp/branches/release-1.0-patches  
U src\Hello.java  
Aktualisiert zu Revision 38.  
  
~/svn-bsp $svn info | grep URL  
URL: file:///c:/programme/subversion/svnrepo/svn-bsp/branches/release-1.0-patches
```

42

Bugfix Release aus Branch erzeugen



43

Bugfix im Branch ausführen und releasen

- Gleiche Vorgehensweise wie zuvor
 - Unterschied ist Repository Verzeichnis der Arbeitskopie

```
~/svn-bsp $svn info | grep URL
URL: file:///c:/programme/subversion/svnrepo/svn-bsp/branches/release-1.0-
patches

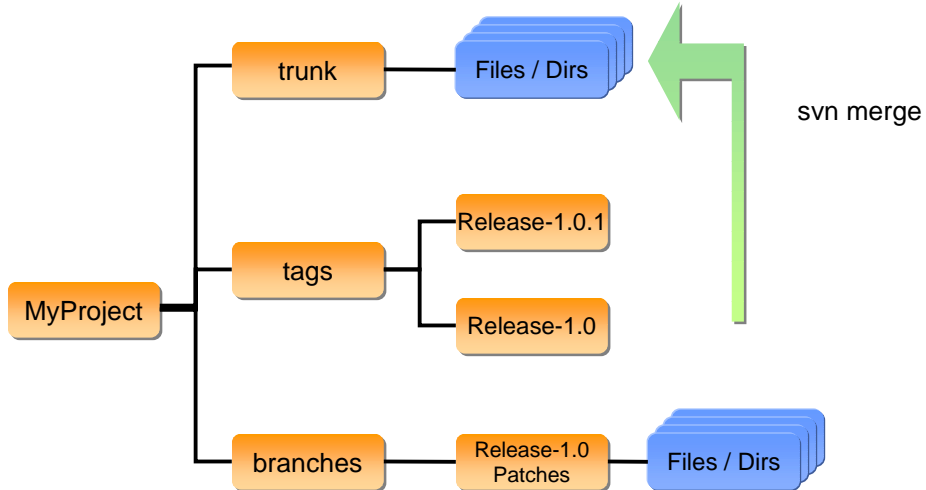
~/svn-bsp/src $svn ci -m "Bugfix in Methode foo()" Hello.java
Sende      Hello.java
Übertrage Daten .
Revision 39 übertragen.

~/svn-bsp $svn copy . $SVNROOT/svn-bsp/tags/release-1.0.1 \
-m "Created Release 1.0.1"

Revision 40 übertragen.
```

44

Bugfixes in Hauptentwicklung übernehmen



45

Merge vorbereiten

- Durch Subcommand `log` ermitteln wann Branch stattgefunden hat
 - Switch „--stop-on-copy“ einsetzen
- Lokale Arbeitskopie durch `switch` wieder auf Trunk umsetzen

```
~/svn-bsp $svn log --stop-on-copy $SVNROOT/svn-bsp/branches/release-1.0-patches
```

```
-----  
r39 | schluffi | 2005-04-22 12:08:53 +0200 (Fr, 22 Apr 2005) | 1 line
```

```
Bugfix in Methode foo()
```

```
-----  
r38 | schluffi | 2005-04-22 10:43:46 +0200 (Fr, 22 Apr 2005) | 1 line
```

```
Creating a branch for Release 1.0  
-----
```

```
~/svn-bsp $svn switch $SVNROOT/svn-bsp/trunk
```

```
U src\Hello.java  
Aktualisiert zu Revision 40.
```

46

Merge durchführen



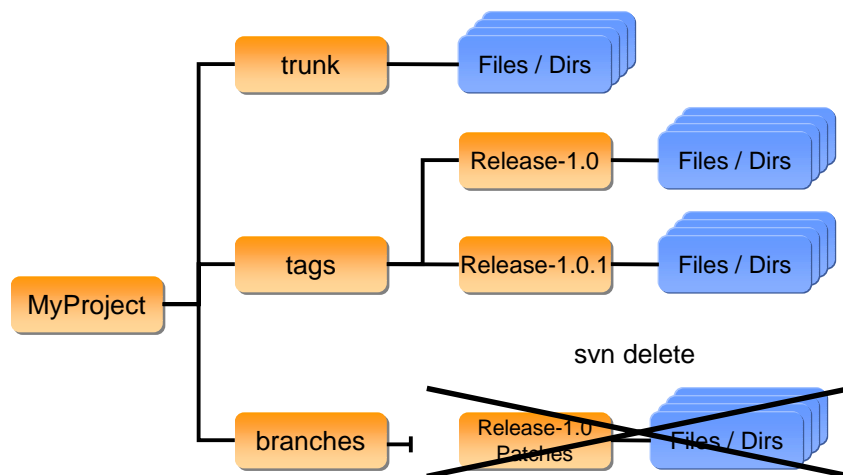
- Änderungen im Branch durch `merge` auf Trunk anwenden
 - Anschließend als neue Revisionen einchecken
- Am Merge beteiligte Revisionen in Lognachricht hinterlegen
 - Um Konflikte bei mehrmaligem Mergen vermeiden zu können

```
~/svn-bsp $svn merge -r38:39 $SVNROOT/svn-bsp/branches/release-1.0-patches
U src\Hello.java

~/svn-bsp $svn ci -m "Merged Release 1.0 patches r38:39 into trunk" .
Sende          src\Hello.java
Übertrage Daten .
Revision 41 übertragen.
```

47

Nicht mehr benötigten Branch löschen



48

Gliederung



- Subversion Grundlagen
- Dateien unter Subversion Kontrolle stellen
- Mit Dateien in Subversion arbeiten
- Markierungen und Zweige in Subversion
- **Properties**
- Zusammenfassung

49

Properties



- Metadaten für Dateien und Verzeichnisse
 - Versionierte Name-Werte-Paare
 - Neues Feature in Subversion
- Ergänzung zu eigentlicher Arbeit mit Suversion
 - Subcommands `propdel`, `propedit`, `propget`, `proplist`, `propset`
 - Automatisches Setzen über Runtime Configuration Area möglich

50

SVN Properties



- Metadaten mit spezieller Bedeutung für Subversion
 - Beginnen mit reserviertem Prefix svn:
- Ergänzen Subversion um weitere Funktionalität
 - Teilweise CVS Funktionalität (Keywords, .cvsignore)
- Bearbeitung wie normale Properties
 - Automatisches Setzen über Runtime Configuration Area, again!

51

Subversion Keywords Property



```
~/svn-bsp/src $svn propset svn:keywords "Id" Hello.java
Eigenschaft 'svn:keywords' für 'Hello.java' gesetzt

~/svn-bsp/src $svn ci -m "Added keywords property" Hello.java
Sende      Hello.java

Revision 46 übertragen.

~/svn-bsp/src $svn update
Revision 46.
```

52

Keywords Property Auswirkung

```
// $Id$  
public class Hello {  
    public static void main (String[] args) {  
        System.out.println("-----");  
        System.out.println("Hello World");  
        System.out.println("-----");  
    }  
    public String foo(){  
        return "Fancy feature";  
    }  
    public String oneTrueEditor(){  
        return "XEmacs";  
    }  
}
```

```
// $Id: Hello.java 46 2005-04-25 11:50:24Z schluffi $  
public class Hello {  
    public static void main (String[] args) {  
        System.out.println("-----");  
        System.out.println("Hello World");  
        System.out.println("-----");  
    }  
    public String foo(){  
        return "Fancy feature";  
    }  
    public String oneTrueEditor(){  
        return "XEmacs";  
    }  
}
```

53

Dateien ignorieren

- Bestimmte Ressourcen nicht in Subversion verwalten
 - Selbst erzeugte Compile und Bibliotheken (*.class, *.obj, *.jar, *.lib)
 - Logging und Debug Output
 - Automatisch erzeugte Dokumentation (Javadoc Ausgabe)
- Generell alles, was sich automatisiert erzeugt läßt
 - Restliche Ressourcen unbedingt mitverwalten
 - Buildskripte (build.xml, Makefile, etc.)
- Steuerung über svn:ignore Property je Verzeichnis
 - Auch über global-ignores in Runtime Configuration Area

54

Subversion Ignore Property



```
~/svn-bsp/src $svn status
?      Hello.class

~/svn-bsp/src $svn propset svn:ignore '*.class' .
Eigenschaft 'svn:ignore' für '.' gesetzt

~/svn-bsp/src $svn ci -m "Added keywords property" .
Sende      src

Revision 47 übertragen.

~/svn-bsp/src $svn status

~/svn-bsp/src $svn --no-ignore status
I      Hello.class
```

55

Gliederung



- Subversion Grundlagen
- Dateien unter Subversion Kontrolle stellen
- Mit Dateien in Subversion arbeiten
- Markierungen und Zweige in Subversion
- Properties
- **Zusammenfassung**

56

Wichtigste Unterschiede zu CVS auf einen Blick



- Revisionen nicht mehr auf Dateien sondern auf Verzeichnisbaum
- Versionierung von Verzeichnisbäumen nicht Dateinhalten
- Verzicht auf Sonderstellung von Branches und Tags
- Properties für Metadaten für Dateien und Verzeichnisse
- Konfliktbereinigung ist anwenderfreundlicher
- Spezielle Auslegung auf Reduzierung von Netzwerklast

57

Fazit



- Pro
 - Subversion Arbeitsweise konzeptionell de-facto identisch zu CVS
 - Subversion bringt viele kleine Verbesserungen
 - Subversion behebt einige deutliche CVS Schwächen
 - Subversion wird aktiver entwickelt als CVS
 - Gute Admintools und Scriptingunterstützung
- Contra
 - Konzeptioneller Sprung beim Einstieg (Verzeichnisbaum Revisionen)
 - Verzicht auf Branches und Tags ist umstritten
 - Graphische Clients zum Teil noch nicht die Güte der CVS Clients
 - Verbesserungen teils transparent durch vorhandene CVS Clients

58

If you remember one thing

„Subversion tut das Gleiche wie CVS ist aber gegenüber CVS deutlich verbessert.“

(SuSE 9.3 - Beschreibung zu Subversion)

59

Literaturhinweise



- Version Control with Subversion
 - Sprache: Englisch
 - Broschiert - 299 Seiten - O'Reilly & Associates
 - Erscheinungsdatum: 1. Juni 2004
 - ISBN: 0596004486

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

60

Links



- Subversion Homepage
 - <http://subversion.tigris.org>
- Version Control with Subversion Book
 - <http://svnbook.red-bean.com>
- TortoiseSVN Windows Client
 - <http://tortoisesvn.tigris.org>
- Subclipse Eclipse Client
 - <http://subclipse.tigris.org>
- cvs2svn Homepage
 - <http://cvs2svn.tigris.org/>

61



Fragen ?

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Version: 1.0



**Vielen Dank für Ihre
Aufmerksamkeit !**

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Version: 1.0