

Thema, Ziele: Versionsverwaltung mit Subversion (SVN) - Fortsetzung

Aufgabe 1: Subversion und Konflikte – Projekt "Josufu"

1. Einführung, Ausgangslage

Im Folgenden wird davon ausgegangen, dass am Projekt "Josufu" zwei Personen mitarbeiten, nämlich "Sally" und "Mike".

Sie spielen nun abwechselungsweise die Rolle von Sally und Mike. Da beim Einchecken (commit) automatisch immer ihr richtiger Username (also nicht "Sally" oder "Mike") verwendet wird, geben Sie jeweils beim Commit im Kommentar an, ob dieses von Sally oder Mike durchgeführt wird. ("Im wahren Leben" ist das natürlich nicht nötig.)

2. Sally: Repository und eigenes Arbeitsverzeichnis erstellen

Sie sind *Sally*. Erstellen Sie, wie in der letzten Übung bei Aufgabe 2, ein neues (lokales) SVN-Repository "svn-repos_Josufu" in Ihrem Verzeichnis "svn-root".

Schreiben Sie anschliessend die übliche Grundstruktur in dieses Repository. Tipp: "import (to ...)" (siehe letzte Übung, Aufgabe 2.2). – Diese Grundstruktur finden Sie als Vorlage zu dieser Übung im Unterverzeichnis "Vorlage_ueb10_svn2/svn_grundstruktur".

Erstellen Sie nun ein SVN-Arbeitsverzeichnis "work_sally". Checken Sie dazu den "trunk" (nur den "trunk!") aus dem Josufu-Repository aus.

3. Sally: Neue Dateien "number.txt" und "day.txt", diese einchecken

Sie sind immer noch *Sally*: Auf dem Skript-Server finden Sie als Vorlage zu dieser Übung im Unterverzeichnis "Vorlage_ueb10_svn2/tmpdir" die Dateien "number.txt" und "day.txt". Kopieren Sie diese in ihr Arbeitsverzeichnis "work_sally", stellen Sie sie unter SVN-Verwaltung ("svn add") und checken Sie diese anschliessend sofort ein. Verwenden Sie dabei einen passenden Kommentar, wie etwa "Sally: Erste Dateien hinzugefügt."

Kontrollieren Sie danach, ob diese zwei Dateien in Ihrem Repository "svn-repos_Josufu" wirklich im "trunk" vorhanden sind.

Führen Sie danach noch ein "update" aus.

4. Mike arbeitet neu am Projekt mit

Sie sind nun *Mike*: Erstellen Sie (als Mike) ein SVN-Arbeitsverzeichnis "work_mike". Checken Sie dazu den "trunk" (nur den "trunk!") aus dem Josufu-Repository aus.

Sally und Mike haben nun identische Arbeitskopien.

5. Mike ändert "number.txt"

Sie sind immer noch *Mike*: Fügen Sie in "work_mike/number.txt" die beiden Zeilen "five" und "six" hinzu.

Checken Sie diese Änderung ins Repository ein.

→ Im FE "work_mike" anwählen → SVN Commit → Kommentar: "mike: neue Zahlen".

6. Sally prüft ob im Repository etwas geändert hat

Sie sind *Sally*: Sie möchten kontrollieren ob Ihre Arbeitskopie noch aktuell ist bzw. anders formuliert, ob im Repository inzwischen etwas geändert hat.

→ Im FE "work_sally" anwählen → TortoiseSVN → Check for Modification → Check repository.

Differenz feststellen:

→ Im FE "number.txt" (in "work_sally") anwählen → TortoiseSVN → Show log
→ "number.txt" zuoberst anwählen → KM: "Compare with Base".

7. Sally: "mergen" (zusammenführen) – trotz Unterschieden

Sie sind Sally: → Im FE "work_sally" anwählen → update

"number.txt" von Sally enthält jetzt neu auch die Zeilen "five" und "six". → Sally und Mike sind jetzt beide auf demselben Stand!

8. Sally und Mike: Gleichzeitige Änderungen (*nicht* überlappend) → Kollision

Sally und Mike ändern beide "number.txt":

Sie sind Sally: Ändern Sie "six" nach "SIX" (im Verzeichnis "work_sally").

Sie sind Mike: Ändern Sie "zero" nach "ZERO" (im Verzeichnis "work_mike").

- Sally checkt ihre Version *zuerst* ein:

Sie sind Sally: → FE: "work_sally" → "SVN commit" (Kommentar: "Sally: SIX ist wichtig")

- Mike checkt auch seine Version ein:

Sie sind Mike: → FE: "work_mike" → "SVN commit" (Kommentar: "Mike: ZERO betonen")


→ Meldung (rot): **Error: Commit failed: File '/trunk/number.txt' is out of date.**

Was jetzt?

- Mike aktualisiert seine Version!


Sie sind immer noch Mike: → FE: "work_mike" → "SVN update".

→ Meldung (grün): **Merged ... work_mike\number.txt.**

aber: Anzeigen bei "work_mike" und "number.txt" im FE sind immer noch  (rot).

Grund: Sallys Änderungen wurden zwar in Mikes "number.txt" integriert, aber noch nicht ins Repository eingchecked

→ FE: "work_mike" → "SVN commit" (Kommentar: "Mike: ZERO betonen")

→ Anzeigen in FE sind neu  (grün) bei "number.txt" und "work_mike".

9. Gleichzeitige Änderungen, *überlappend* → Konflikt

Sally und Mike aktualisieren beide ihre Arbeitsverzeichnisse:

→ tun Sie das! → Sally und Mike sind nun beide auf demselben Stand.

- Sally und Mike ändern beide "number.txt":

Sie sind Mike: Ändern Sie die zweite Zeile von "one" zu "ichi" (im Verzeichnis "work_mike").

Sie sind Sally: Ändern Sie die zweite Zeile von "one" zu "uno" (im Verzeichnis "work_sally").

- Sally checkt ihre Version *zuerst* ein:

Sie sind Sally: → FE: "work_sally" → "SVN commit" (Kommentar: "Sally: italienisch ist besser")

- Mike checkt auch seine Version ein:

Sie sind Mike: → FE: "work_mike" → "commit" (Kommentar: "Mike: Kunde wünscht Japanisch")


→ Meldung (rot): **Error: Commit failed: File '/trunk/number.txt' is out of date.**


Was jetzt? Gleiche Meldung wie vorhin, also gleiche Reaktion wie vorhin:

- Mike aktualisiert seine Version!



Sie sind immer noch Mike: → FE: "work_mike" → "SVN update".

→ Meldung (rot): **Conflicted ... work_mike\number.txt.**

→ Anzeige im FE ist neu  (orange) bei "number.txt" und "work_mike".

"number.txt" (mit ) enthält neu "beide" Versionen, in "work_mike" sind drei neue "number.txt.###" Dateien entstanden.

→ Betrachten Sie diese Dateien mit dem Editor. Was ist ihr Inhalt?

- **Konflikt:** *Es kann nur eine Variante richtig sein!*
- Mike spricht mit Sally fragt anschliessend beim Kunden nach: Richtig ist "one" in Japanisch und "two" in italienisch.
- Sie sind *Mike*: korrigieren Sie "number.txt" entsprechend: "ichi" für "one", "due" für "two".
- Subversion darüber informieren, dass (in "work_mike") der Konflikt gelöst ist:
 - FE: "work_mike" → "SVNTortoise" → "resolved".
 - Anzeige im FE ist neu  (rot) bei "number.txt" und "work_mike".
Grund: Version von Mike wurde zwar angepasst aber noch nicht eingchecked.
 - FE: "work_mike" → "SVN commit" (Kommentar: "Mike: 'ichi' für 'one', 'due' für 'two'. ")
 - Anzeigen im FE sind neu  (grün) bei "number.txt" und "work_mike".

Aufgabe 2: Subversion und Eclipse

Einführung

Für Eclipse gibt es zwei Subversion-Plugins: "Subclipse" und "Subversive". Wir verwenden "Subclipse".

Falls das Subclipse-Plugin noch nicht installiert ist: Im Eclipse: "Help" → "Eclipse Marketplace..." → "Search":
Find: Subclipse ...

1. Daten aus einem bestehendem SVN-Repository Daten in ein Eclipse-Projekt ausschecken

Erstellen Sie wie folgt das Eclipse-Projekt "MyJosufu" mit den Dateien Ihres lokalen SVN-Repository "svn-repos_Josufu";

File → New → Project ... ; SVN → "Checkout Projects from SVN" → NEXT → "Create a new repository location" → NEXT; (kopieren Sie den Pfad aus dem Repro-Browser: file:///...) ... NEXT; "Check out as project in the workspace, Project Name 'MyJosufu' ".

Beachten Sie, dass im Eclipse-Project-Explorer die ausgescheckten Dateien mit ihrer "Revision-Number" und einem zusätzlichen Symbol versehen sind.

Ersetzen Sie mit Eclipse in der Datei "number.txt" "three" durch "trois" und checken Sie diese Änderung ein ("commit", Kommentar: "neu französisch"). – Sie finden dazu im Eclipse KM der Datei / des Projekts den Eintrag "Team" mit einem ähnlichen Menu wie Sie es schon von TortoiseSVN her kennen.

Erzeugen Sie eine neue Datei "month.txt" in "MyJosufu" und stellen Sie diese unter SVN-Verwaltung.

Allgemein: "Spielen Sie" mit Subversion unter Verwendung von Eclipse.

2. Ein bestehendes Projekt in ein (bestehendes) SVN-Repository einchecken

Wählen Sie irgendein Eclipse-Projekt von früher aus, das unter SVN-Verwaltung gestellt werden soll.

Erstellen Sie nun mit TortoiseSVN ein neues (lokales) SVN-Repository für dieses Projekt (Eclipse kann das nicht!).

Mit dem Kontextmenü "Team" → "Share Project..." → "SVN" → kann nun das Eclipse-Projekt in dieses Repository eingchecked werden. Tun Sie das.

Aufgabe 3: SVN-Repository gemeinsam für Projekt "Cusep" verwenden

1. Projektteam bilden

Suchen Sie sich ein oder zwei Mitstudierende mit denen Sie das Projekt "Cusep" in Zukunft gemeinsam weiterentwickeln möchten. Das so gebildete Projektteam darf maximal aus 3 Personen bestehen. Bestimmen Sie einen Projektleiter für ihr Team.

2. Ein Repository gemeinsam nutzen

Auf dem Skript-Server finden Sie unter ".../PmSwEng/Anleitungen" die Datei "HSR-Subversion-HowTo.pdf".

Darin wird erklärt, wie Sie ein SVN-Repository erstellen und nutzen können, dass gleichzeitig von mehreren Personen via Internet zugänglich ist. – Beachten Sie, ausserhalb der HSR müssen Sie dazu eine VPN-Verbindung zur HSR aufgebaut haben.

Studieren Sie diese Anleitung. Erstellen Sie entsprechend für das Projekt "Cusep" ein SVN-Repository mit dem Projektnamen "CusepXXXX". (Ersetzen Sie XXXX durch ein paar Buchstaben Ihrer Wahl. Damit soll erreicht werden, dass alle Projekte aus Ihrer Klasse unterschiedliche Namen haben.) Richten Sie danach für jedes Mitglied Ihres Teams einen Benutzer ein.

Melden Sie sich anschliessend über "svn://svns.hsr.ch/<Projektname>" an und checken Sie danach eine erste Version in dieses Repository ein.

Verteilen Sie die folgenden Arbeiten 3 und 4 im Team, sprechen Sie untereinander ab!

Arbeiten Sie parallel. Checken Sie regelmässig Ihre geänderte Arbeitskopie wieder ein ("commit"), damit, falls mal etwas schiefgeht, sie auf eine frühere Version zurückgreifen können.

3. Projekt "Cusep" konsolidieren ("aufräumen")

Räumen Sie nun ihr Projekt "Cusep" gründlich auf. Entfernen Sie dabei die Tests mit den gewöhnlichen "assert"-Anweisungen im "main()".

4. "MyDate" erweitern

Wäre es nicht schön, den zu einem Datum gehörenden Wochentag erfahren zu können? Wissen Sie, an welchem Wochentag Sie geboren wurden?

Mit Hilfe der `daysSince1()`-Methode (und dem Modulo-Operator) ist das nun einfach herauszufinden!

Erweitern Sie Ihre Klasse `MyDate` um eine entsprechende Memberfunktion `getWeekday()`, welche 0 für Mo, 1 für Di, ..., 6 für So zurückgibt.

Beispiel-Aufruf:

```
MyDate dl(4, 5, 2012);  
assert (dl.getWeekday() == 4);
```

Anmerkung zur "Parallelarbeit" in Projekten

In einem „echten“ Projekt werden Sie Ihre Änderungen erst im "trunk" einchecken, wenn Ihr Code in einem einigermaßen stabilen Zustand ist. Unfertiger (fehlerhafter) Code könnte sonst bewirken, dass ein Update der Teamkollegen zu einer nicht-funktionierenden oder gar nicht-kompilierenden Version führt.

Bis der von Ihnen bearbeitete Code genügend stabil ist können Sie die Dateien z.B. in einem eigenen Branch (unter "branches") verwalten.