Podstawy pracy z wersjonowanymi plikami

Part I
Podstawy

1 Tworzenie/Dodawanie repozytorium

1.1 Repozytorium lokalne

W celu dydaktycznym utworzymy lokalne repozytorium SVN (nie bedziemy omawiac tego procesu, pozniej bedziemy pracowac na serwerze internetowym):

```
cd /tmp
svnadmin create svn_test_server
mkdir repos
cd repos
svn checkout file:///tmp/svn_test_server test1
svn checkout file:///tmp/svn_test_server test2
```

2 Pobranie zmian wprowadzonych przez innych użytkowników

Jeśli pracujemy wraz z innymi zaktualizowanie plików w naszej kopii roboczej jest już koniecznością. Choćby po to by nie pracować nad rzeczami, które ktoś mógł już zrobić. Wykonujemy to za pomocą komendy:

```
svn update
```

Nie musimy już wpisywać adresu repozytorium - wszystkie potrzebne dane przechowywane sa w ukrytych katalogach .svn Po wykonaniu tej komendy wyświetlone zostaną wszystkie zmiany w plikach w postaci:

```
svn update
A blabla.py
D foofoo.py
U barbar.py
C jakis.py
```

```
G zupelnieinny.py
Updated to revision 48.
```

Gdzie A oznacza dodany plik, D - skasowany, U - zmieniony, C - pliki w konflikcie (gdy pobrane zmiany dotyczą tego samego fragmentu pliku, który był edytowany lokalnie), G - połączony za pomocą komendy merge z innej linii rozwojowej. Dodatkowo otrzymujemy informację o numerze rewizji repozytorium.

3 Wprowadzanie zmian w plikach i folderach

3.1 Dodanie pliku

Pliki dodajemy za pomocą polecenia:

```
svn add [plik(i) lub folder(y)]
```

Po wykonaniu tej komendy pliki zostaną oznaczone jako oczekujące na dodanie. Rzeczywiste dodanie nastąpi w momencie najbliższego commitu. Jeśli wskazany został folder cała jego zawartość również zostanie dodana do repozytorium. Foldery możemy także utworzyć za pomocą polecenia svn mkdir [folder] Jeśli utworzymy pliki normalnymi poleceniami systemu i nie dodamy ich za pomocą svn add nie będą wersjonowane!

3.2 Usunięcie pliku

Pliki usuwamy za pomocą:

```
svn delete [plik(i) lub folder(y)]
```

Jeśli wybrany został plik natychmiast zniknie on z kopii roboczej, jeśli folder to zostanie on (i jego zawartość) oznaczone jako do usunięcia. Rzeczywiste usunięcie plików z repozytorium nastąpi, podobnie jak w przypadku svn add dopiero przy najbliższym commicie. Jeśli skasujemy plik normalnymi poleceniami systemu a nie za pomocą svn delete zostaną ponownie pobrane z repozytorium przy najbliższym update. Warto o tym pamiętać - jeśli coś solidnie popsuliśmy wystarczy skasować "popsuty" plik lub folder i wykonać update by móc cieszyć się poprzednią wersją pliku.

3.3 Przenoszenie i kopiowanie plików

Wykonujemy za pomocą poleceń

```
svn move [zródlo] [cel]
svn copy [zródlo] [cel]
```

Co najważeniejsze SVN "pamięta" całą historię kopiowanego pliku. Podobnie jak w przypadku dodawania i kasowania plików trzeba wykonać commit by zmiany zostały zapisane w repozytorium.

3.4 Dokonywanie zmian w plikach

W wypadku zwykłych zmian w plikach nie musisz używać żadnych specjalnych komend Subversion. Po prostu edytujesz pliki w swoim ulubionym edytorze.

4 Sprawdzenie jakie zmiany zostały dokonane

Przed wysłaniem wyniku swojej pracy do repozytorium warto sprawdzić co tak na prawdę zmieniliśmy. Co najważniejsze wszystkie przedstawione w tym punkcie komendy nie wymagają połączenia z repozytorium. Wszystkie potrzebne dane znajdują się w katalogach .svn Do tego celu możemy wykorzystać:

4.1 Wykaz zmian

Wykaz wszystkich zmienionych plików i folderów otrzymamy za pomocą polecenia:

```
svn status
```

Zwrócona zostanie lista plików poprzedzona sześcioma kolumnami informującymi o rodzaju zmian. Nas na razie interesuje jedynie pierwsza kolumna zawierająca oznaczenia podobne do tych poznanych w komendzie svn update I tak: A - element przygotowany do dodania, D - element przygotowany do skasowania, U - element zmodyfikowany, R - zamieniony (element został przygotowany do skasowania a następnie został utworzony nowy o tej samej nazwie), ? - element nie podlegający wersjonowaniu (na przykład plik dodany normalnymi metodami a nie za pomocą svn add), ! - brakujący element (na przykład skasowany normalnymi metodami a nie za pomocą svn delete)

4.2 Wyszczególnienie zmian w pliku

Czasem może się zdarzyć, że nie pamiętamy jakich zmian dokonaliśmy w pliku. Pomocne wtedy jest polecenie:

```
svn diff
```

Zwraca ono listę zmian w plikach w formacie unified diff format

4.3 Cofnięcie zmian

Dopóki nie dokonaliśmy commitu możemy łatwo cofnąć wprowadzone w kopii roboczej zmiany. Używamy polecenia:

```
svn revert [nazwa pliku]
```

Polecenie bardzo przydatne jeśli przez pomyłkę skasowaliśmy jakiś plik lub rozmyśliliśmy się jeśli chodzi o zmiany.

5 Pobranie aktualnych wersji plików

Po raz kolejny wykonujemy update by sprawdzić czy nikt inny nie dokonał zmian w tych samych plikach co my. W większości wypadków Subversion bez problemu poradzi sobie z wprowadzeniem odpowiednich zmian. Problem pojawi się gdy okaże się, że ktoś edytował ten sam fragment pliku co my. Subversion nie jest w stanie samemu zaktualizować pliku i pojawia się konflikt (pamiętacie literkę 'C' w svn update?)

6 Rozwiązywanie konfliktów

W momencie powstania konfliktu Subversion tworzy 3 nowe pliki i modyfikuje plik będący w konflikcie. Trzy nowe pliki to:

- plik.mine - wersja pliku z naszymi zmianami
- plik.rOLDREV - gdzie OLDREV to numer rewizji sprzed naszych zmian
- plik.rNEWREV - gdzie NEWREV to numer rewizji aktualnej wersji w repozytorium (ze zmianami wprowadzonymi przez kogoś innego)

W pliku będącym w konflikcie dopisane zostają dodatkowe linijki z zaznaczonym konflikotwym fragmentem w formie:

```
linia niezmieniona
<<<<<<< .mine
linia zmieniona przeze mnie
======= linia zmieniona przez kogos
>>>>>>> .rNEWREV linie niezmieniona
```

Po skontaktowaniu się z osobą, która wprowadziła swoje zmiany poprawiamy plik by zawierał ostateczną wersję (edytując go odpowiednio lub używając jednego z 3 plików jako wzoru) i wykonujemy polecenie:

```
svn resolved [nazwa pliku]
```

Subversion przygotuje plik i konflikt zostanie rozwiązany przy najbliższym commicie. Przy okazji skasowane zostaną trzy dodatkowe pliki (.mine, rOLDREV, .rNEWREV)

7 Wysłanie zmian do repozytorium

Gdy mamy już aktualne wersje plików i nie ma konfliktów wreszcie możemy wysłać zmiany do repozytorium za pomocą polecenia:

```
svn commit
```

Przy wysyłaniu commitu musimy dodać komentarz z opisem, który zostanie dołączony do logów. Jeśli użyjemy polecenia commit w formie jak powyżej

otworzy się domyślny edytor tekstu, w którym możemy wpisać komentarz. Możemy również użyć polecenie w formie: svn commit -m "treść komentarza" Rzecz jasna by wykonać to polecenie musimy nawiązać połączenie z repozytorium.

Part II
Cwiczenia

8 Lokalne

1. Utworz nastepujace katalogi i pliki w “working copy“ test1 (mkdir, touch):

   ```
   doc
   src
   doc/a.txt
   doc/b.txt
   ```

2. Dodaj nastepujace katalogi i pliki do “working copy“ test1 (svn add):

   ```
   doc
   doc/a.txt
   ```

3. W “working copy” test1: w pliku a.txt utworz tekst “Who is he?\nHis name is x\nWho knows...” (uzyj ulubionego edytora tekstowego; \n oznacza Enter).

4. W “working copy” test1: zobacz jakie zmiany zostana wprowadzone do repozytorium (svn diff).

5. Wyslij zmiany z “working copy” test1 do repozytorium (svn commit). Nie zapomnij podac komentarze okreslajacego co zmieniles.

6. Przejdz do “working copy” test2. Wykonaj update (cd, svn update). Powinienes zobaczyc wszystkie zmiany (wprowadzone w “working copy” test1).

7. W “working copy” test2: zmien tekst w pliku a.txt na “Who is he?\nHis name is y\nWho knows...”. Stworz rowniez plik “doc/c.txt” i dodaj go do “working copy” (svn add). Nie wprowadzaj zmian do repozytorium (tzn. nie wydawaj polecenia svn commit)

8. W “working copy” test1: zmien tekst w pliku a.txt na “Who is he?\nHis name is z\nWho knows...”.

9. W “working copy” test1: wprowadz zmiany do repozytorium (svn commit).

10. W "working copy" test2: wprowadz zmiany do repozytorium (svn commit). Spowoduje to konfilkt (powstana trzy dodatkowe pliki).

11. W "working copy" test2: edytuj plik a.txt. Zdecyduj sie na wersje "Who is he?\nHis name is z\nWho knows...". Zatwierdz rozwiazanie konfliktu (svn resolved doc/a.txt). Wprowadz zmiany do repozytorium (svn commit).

9 Serwer STSE

1. Zanim zaczniemy korzystac z repozytorium zdalnego nalezy przejsc do wybranego katalogu na swoim dysku i pobrac zawartosc zdalnego repozytorium (prosze zastapic user:pass swoimi danymi z serwera sourceforge.net; prosze uwazac na "zlamane linie \" - cala komenda to jedna linia; prosze uwazac przy kopiowaniu adresu z PDF - nie wszystkie znaki sie dobrze przenosza - najlepiej wpisac te linie "recznie"):

   ```
   svn checkout \
   https://user:pass@stse.svn.source\
   forge.net/svnroot/stse/branches/wput-tp/test
   ```

   Od tego momementu mozna zmiany ktore bedziemy wprowadzac lokalnie beda mogly byc w prosty sposob wysylane na serwer na ktorym znajduje sie srodowisko STSE.

2. Prosze odnalezc program swojej grupy w stse/branches/wput-tp/test/tp/ (przy pobraniu repozytorium instrukcja powyzej beda Panstwo mieli tylko katalog tp; przy pobraniu pelnego srodowiska instrukcja w pkt. 4 beda Panstwo mieli pelna sciezke). Prosze utworzyc plik authors.txt ktory zawiera imiona i nazwiska autorow. Prosze dodac ten plik do repozytorium (svn add) i wprowadzic zmiany (svn commit). Prosze nie zapomniec o komentarzach!

3. Prosze przejsc do folderu lustrzanej grupy i stworzyc plik start.py. Plik ten powinien zostac dodany do repozytorium i po wywolaniu:

   ```
   ipython start.py
   ```

   Plik ten powinien uruchamiac program grupy przeciwnej wyswietlajac uprzednio autorow z pliku authors.txt (prosze napisac taki program pythona).

4. W domu prosze sobie zainstalowac pelne srodowisko STSE (bedziemy pracowac w galezi repozytorium):

   ```
   svn checkout \
   https://user:pass@stse.svn.source\
   forge.net/svnroot/stse/branches/wput-tp
   ```