TasPython

Εκμαθηση του Mercurial Βημα Βημα

Εισαγωγή στο Mercurial

Συγγραφέας:
Κωνσταντίνος Αραβανής
e-mail: arabanis@ceid.upatras.gr

Ομάδα:
TasPython

23 Οκτωβρίου 2009

1 Εισαγωγή

Ο παρόν οδηγός αποτελεί ένα εισαγωγικό βήμα για την χρήση του Mercurial και για πιο προχωρημένα θέματα θα πρέπει να απευθυνθείτε σε άλλα σημεία.

Το *Mercurial* πρόκειται για ένα *σύστημα διαχείρισης εκδόσεων* (version control system) το οποίο είναι γραμμένο σε *Python*. Συγκεκριμένα μας βοηθάει να κρατάμε *εκδόσεις* (ή και backups) της δουλείας μας εύκολα με αυτόματο τρόπο χωρίς να χρειάζεται να δημιουργούμε ως συνήθως πολλούς φακέλους με ονόματα όπως: version-1, version-2, ..., version-∞. Αυτό που κάνει το *Mercurial* είναι να ελέγχει την τελευταία έκδοση με την τρέχουσα προς δημιουργία και να κρατά τις διαφορές τους. Επίσης πολλές φορές σε ένα κώδικα ή ένα αρχείο μπορεί να δουλεύουν ταυτόχρονα πολλά άτομα. Το *Mercurial* είναι ένα εργαλείο που μπορεί να μας βοηθήσει στο να *συγχωνεύσουμε αυτόματα* τις αλλαγές που κάνουν διάφορα άτομα πάνω στα ίδια αρχεία και να μπορούμε να κάνουμε εναλλαγές στο κώδικα από την μία έκδοση στην άλλη πολύ εύκολα.

2 Εγκατάσταση

1. Για *Linux* υπάρχει ήδη έτοιμο πακέτο σχεδόν για όλες τις distributions

2. Για *Mac OS X* είτε πρέπει να γίνει compile ο κώδικας του *Mercurial*, είτε να εγκατασταθεί το εκτελέσιμο.

3. Για *Windows* προτείνεται το *TortoiseHg* που μπορείτε να το βρείτε εδώ: `http://mercurial.selenic.com/wiki/TortoiseHg`

Για περισσότερες πληροφορίες σχετικά με τα παραπάνω επισκεφτείτε αυτή τη σελίδα: `http://mercurial.selenic.com/wiki/Download`

3 Ορίζοντας το username σας

Ως προεπιλογή το *Mercurial* χρησιμοποιεί ως όνομα χρήστη κάτι του τύπου '*me@my_pc*' για τα *commits*, κάτι το οποίο είναι ανούσιο. Προτείνεται ο χρήστης να ρυθμίσει κατάλληλα το */.hgrc* (ή σε Windows το *%USERPROFILE%/Mercurial.ini*) ώστε να εμφανίζει το όνομα του και το e-mail του, προσθέτοντας τις παρακάτω γραμμες:

```
[ui]
username = Kostantinos Aravanis <arabanis@ceid.upatras.gr>
```

4 Βοήθεια! - *hg*[1] *help*

Όπως είναι λογικό και όπως κάνατε σε κάθε καινούργιο σύστημα, κάπου θα πρέπει να απευθυνθείτε για βοήθεια. Το *Mercurial* έχει τη δικιά του εντολή βοήθειας που σας επιτρέπει πατώντας:

```
me@my_pc:~/project_folder$ hg help
```

να δείτε τη λίστα με τις εντολές και τα διάφορα θέματα που μπορεί να σας απασχολούν. Επίσης αν θέλετε να μάθετε, για παράδειγμα, περισσότερα για την εντολή *add* μπορεί να γράψετε:

```
me@my_pc:~/project_folder$ hg help add
```

5 Δημιουργία του *Repository* - *hg init*

Έστω ότι θέλουμε να ξεκινήσουμε ένα project και θέλουμε να διαχειριστούμε τις διάφορες εκδόσεις του με το *Mercurial*, το πρώτο που έχουμε να κάνουμε είναι να αρχικοποιήσουμε ένα *hg repository*[2]. Αυτό το κάνουμε ως εξής:

```
me@my_pc:~$ mkdir project_folder
me@my_pc:~$ cd project_folder/
me@my_pc:~/project_folder$ hg init
```

Επίσης αν θέλετε γενικά το *Mercurial* να αγνοεί στο project σας κάποια αρχεία με συγκεκριμένα *glop patterns* και *regular expressions* αυτό που έχετε να κάνετε είναι να προσθέσετε ένα αρχείο με όνομα *.hgignore* κάτω από το κύριο φάκελο του project (παραπάνω για παράδειγμα αυτός ήταν ο *project_folder*), το οποίο θα περιέχει αυτά που θέλετε να αγνοούνται από το *Mercurial*. Ένα παράδειγμα αυτού του αρχείου θα μπορούσε να είναι το παρακάτω:

```
syntax: glob
*.orig
*.rej
*~
*.o
tests/*.err
```

[1] Το hg πρόκειται για την κωδική ονομασία του στοιχείου Mercurial στον περιοδικό πίνακα.

[2] Με τον όρο repository εδώ εννοούμε κάτι το οποίο παρακολουθεί την πορεία των αλλαγών στο project σε όλο το παρελθόν. Στην ουσία είναι κρυμμένο στον φάκελο *.hg*.

```
syntax: regexp
.*\#.*\#$
```

6 Πρόσθεση Αρχείου στο *Repository - hg add*

Αν στο φάκελο του project σας δημιουργήσατε ένα καινούργιο αρχείο, έστω *new.py* και θέλετε να ληφθεί υπόψη από το *repository* σας, αυτό που έχετε να κάνετε είναι να πατήσετε:

```
me@my_pc:~/project_folder$ hg add new.py
```

Αν πάλι έχετε δημιουργήσει πολλά αρχεία και θέλετε να προστεθούν όλα στο *repository* απλά πατήστε:

```
me@my_pc:~/project_folder$ hg add
```

7 Αφαίρεση Αρχείου από το *Repository - hg remove*

Αν κάποιο αρχείο που χρησιμοποιούσατε σε παλιότερες εκδόσεις δεν το χρειάζεστε πια και θέλετε να το διαγράψετε από το *repository* σας, αυτό που έχετε να κάνετε είναι να πατήσετε:

```
me@my_pc:~/project_folder$ hg remove useless.txt
```

Αυτή η εντολή διαγράφει άμεσα και το αρχείο από το φάκελο του project.

8 Μετονομασία Αρχείου στο *Repository - hg rename*

Στο σημείο αυτό πρέπει να τονιστεί ότι ο σωστός τρόπος για την μετονομασία ενός αρχείου είναι μέσω της *hg rename*. Συγκεκριμένα, έστω ότι θέλετε να μετονομάσετε ένα αρχείο από sth.py σε sth_else.py, αυτό που έχετε να κάνετε είναι να πατήσετε:

```
me@my_pc:~/project_folder$ hg rename sth.py sth_else.py
```

Η παραπάνω εντολή μετονομάζει το αρχείο στο φάκελο στον οποίο δουλεύετε και αυτόματα σημειώνει το παλιό αρχείο προς διαγραφή από το *repository* και το καινούργιο προς πρόσθεση.

Ένας κακός τρόπος για να κάνετε το ίδιο θα ήταν ο ακόλουθος:

```
me@my_pc:~/project_folder$ cp sth.py sth_else.py
me@my_pc:~/project_folder$ hg remove sth.py
me@my_pc:~/project_folder$ hg add sth_else.py
```

Αλλά με αυτό τον τρόπο το *Mercurial* δεν γνωρίζει πια ότι αυτό είναι το ίδιο αρχείο και το αναγκάζετε να χρησιμοποιήσει πολύ παραπάνω χώρο από ότι αν χρησιμοποιούσατε την εντολή *hg rename*.

9 Επίβλεψη της τρέχουσας κατάστασης - *hg status*

Αν τώρα έχετε προχωρήσει σε αλλαγές αρχείων, δημιουργία καινούργιων ή διαγραφή προϋπάρχοντων και θέλετε να μάθετε τι έχει αλλάξει από την τελευταία έκδοση[3] του project σας αρκεί να πατήσετε:

```
me@my_pc:~/project_folder$ hg status
```

Το αποτέλεσμα της παραπάνω εντολής είναι η εμφάνιση μίας λίστας με τα αρχεία που βρίσκονται στο φάκελο και στους υποφακέλους του project σας με ένα χαρακτήρα στην αρχή τους που περιγράφει την κατάσταση τους. Οι δυνατοί χαρακτήρες είναι οι παρακάτω:

?	το παρόν αρχείο δεν έχει προστεθεί στο *repository*
A	το παρόν αρχείο έχει προστεθεί στο *repository* μετά την δημιουργία της τελευταίας έκδοσης του project
M	το παρόν αρχείο έχει προστεθεί στο *repository* αλλά έχει αλλάξει από την τελευταία έκδοση του project
R	το παρόν αρχείο έχει σημειωθεί προς αφαίρεση από το *repository*

Αν πάλι θέλετε να μάθετε για την κατάσταση ενός μόνο αρχείο, για παράδειγμα έστω *file.txt*, όπως είναι λογικό απλά πατάτε:

[3] Για το πως μπορείτε να δημιουργήσετε καινούργιες εκδόσεις μπορείτε να απευθυνθείτε σε παρακάτω ενότητα

```
me@my_pc:~/project_folder$ hg status file.txt
```

Να σημειωθεί ότι η συγκεκριμένη εντολή αγνοεί όλα τα αρχεία που ταιριάζουν στα glob patterns και regular expressions που βρίσκονται στο αρχείο *.hgignore* που αναφέρθηκε και σε παραπάνω ενότητα.

10 Ποίες οι αλλαγές; - *hg diff*

Αν τώρα κατά το *hg status* παρατηρήσατε ότι το *file.txt* παρουσιαζόταν ως αλλαγμένο (M - modified) και θέλετε να δείτε ποιες ακριβώς είναι οι αλλαγές που έχουν γίνει από την τελευταία του έκδοση απλά πατάτε:

```
me@my_pc:~/project_folder$ hg diff
```

Να σημειώσουμε ότι με (−) εμφανίζονται οι γραμμές που αφαιρέθηκαν και με (+) αυτές που προστέθηκαν, οι υπόλοιπες εκτυπώνονται κανονικά.

11 Δημιουργία καινούργιας Έκδοσης - *hg commit*

Τόση ώρα αναφέρονται οι εκδόσεις, αλλά δεν έχει γίνει λόγος για το πως δημιουργείται μία έκδοση. Αυτό είναι απλό! Απλά πατάμε:

```
me@my_pc:~/project_folder$ hg commit -m "comment"
```

Στην ουσία αυτό που γίνεται είναι ότι παγώνουν όλα τα αρχεία που έχουν προστεθεί στο *repository* και αποθηκεύεται η έκδοση με ένα μήνυμα που στη περίπτωσή μας είναι το *comment*.

12 Ποια η Ιστορία; - *hg log*

Αν θέλετε να δείτε την ιστορία του project σας ανάμεσα στις διάφορες εκδόσεις του απλά πατάτε:

```
me@my_pc:~/project_folder$ hg log
```

13 Πότε Γράφτηκε Τι και Από Ποιον - *hg blame*

Αν πατήσετε:

```
me@my_pc:~/project_folder$ hg blame file.txt
```

μπορείτε να δείτε σε ποία έκδοση γράφτηκε η κάθε γραμμή του αρχείου σας.

Ενώ αν πατήσετε:

```
me@my_pc:~/project_folder$ hg blame -u file.txt
```

μπορείτε να δείτε ποιος την έγραψε...!

14 Αλλαγή Έκδοσης - *hg update*

Δύο τρόποι για να 'πλοηγηθούμε' - επιστρέψουμε σε παλιές και άλλες εκδόσεις του project είναι είτε γνωρίζοντας τον αριθμό της έκδοσης (*revision*) που παράγεται αυτόματα από το *Mercurial*, δίνοντας στη πρώτη έκδοση τον αριθμό 0, στη δεύτερη το 1 και ούτω καθ' εξής, είτε με την χρήση των ετικετών (*tags*).

14.1 Με τη χρήση του *revision* αριθμού

Καταρχήν θα πρέπει να σημειωθεί ότι ο *revision* αριθμός πρόκειται για αυτόν που εμφανίζεται δίπλα στο τομέα *changeset* σε κάθε έκδοση όταν πατάμε την εντολή *hg log*.

Για παράδειγμα αν για κάποια έκδοση εμφανίζεται το ακόλουθο:

```
changeset:   1:cfdc8f58ee4c
user:        me@my_pc
date:        Sat Oct 17 19:01:39 2009 +0300
summary:     comment
```

τότε ξέρουμε ότι ο *revision* αριθμός της συγκεκριμένης έκδοσης είναι το 1.

Για να μεταβούμε τώρα σε αυτή την έκδοση αρκεί να πατήσουμε:

```
me@my_pc:~/project_folder$ hg update -r 1
```

14.2 Με τη χρήση των *tags*

14.2.1 Δημιουργία *tags* - *hg tag*

Πριν ξεκινήσουμε να μιλάμε για την αλλαγή εκδόσεων με τη χρήση των *tags* μάλλον θα πρέπει πρώτα να μιλήσουμε για το πως δημιουργούμε *tags* για τις διάφορες εκδόσεις. Στην ουσία τα *tags* είναι ένας πιο εύκολος τρόπος να πλοηγηθούμε στις διάφορες εκδόσεις. Έστω λοιπόν ότι στην έκδοση με *revision* αριθμό 3 θέλουμε να βάλουμε το *tag* 'release 1.0', τότε αυτό που έχουμε να κάνουμε είναι:

```
me@my_pc:~/project_folder$ hg tag -r 3 "release 1.0"
```

Να σημειωθεί ότι κάθε φορά στη τελευταία έκδοση ανατίθεται αυτόματα το tag tip.

14.2.2 Αλλαγή έκδοσης με *tags*

Για να αλλάξουμε έκδοση από αυτή που είμαστε σε κάποια άλλη (υπάρχουσα) με *tag* 'tag_name' απλά πατάμε:

```
me@my_pc:~/project_folder$ hg update tag_name
```

Για να πάμε στη τελευταία έκδοση σύμφωνα και με τα παραπάνω απλά εκτελούμε την εντολή:

```
me@my_pc:~/project_folder$ hg update` tip
```

15 Εύρεση Τρέχουσας Έκδοσης - hg parents

Για να δούμε τα αρχεία του project μας σε ποια έκδοση ανήκουν αρκεί να πατήσουμε την εντολή:

```
me@my_pc:~/project_folder$ hg parents
```

16 Επαναφορά Αρχείου - *hg revert*

Όπως είναι λογικό μπορεί να σας τύχει να κάνετε αλλαγές σε συγκεκριμένα αρχεία, που τελικά να είναι λάθος, και να μη θέλετε να επιστρέψετε ολόκληρο το project σε κάποια παλιότερη έκδοση. Για αυτό υπάρχει μία άλλη εντολή που στόχος της είναι να κάνει επαναφορά συγκεκριμένων αρχείων όπως βρίσκονταν στην έκδοση που βρίσκεστε, όταν αυτή είχε γίνει *commit*. Για να το κάνετε αυτό απλά πατάτε:

```
me@my_pc:~/project_folder$ hg revert file.txt
```

17 Ακύρωση Αποστολής Έκδοσης - *hg rollback*

Αν τώρα χρειαστεί να ακυρώσετε ένα *commit* που έχετε κάνει, αυτό μπορεί να γίνει για το τελευταίο και μόνο *commit*, πατώντας:

```
me@my_pc:~/project_folder$ hg rollback
```

18 Backup του *Repository* - *hg clone*

Για να κάνετε backup του *repository* υπάρχουν δύο τρόποι:

1. Είτε απλά αντιγράφοντας το φάκελο του project σας
2. Είτε ο *προτεινόμενος* που είναι πατώντας την εντολή:

```
me@my_pc:~/project_folder$ cd ..
me@my_pc:~$ hg clone project_folder backup_of_project_folder
```

Η εντολή αυτή παράγει ένα πιστό αντίγραφο του φακέλου (με όλα τα περιεχόμενα του) *project_folder*!

19 Πολλαπλά *Repository*

Υπάρχουν περιπτώσεις που ένα project το αναπτύσσουν πολλά άτομα, που το καθένα έχει αναλάβει ένα συγκεκριμένο κομμάτι. Σε τέτοιες περιπτώσεις τα πολλαπλά *repositories* έρχονται να δώσουν λύση στα προβλήματα διαχείρισής της όλης εργασίας. Συγκεκριμένα αφού δημιουργηθεί ένα κύριο *repository* μετά οι υπόλοιποι εργαζόμενοι πάνω στο project δημιουργούν από ένα κλώνο και κάνουν τις αλλαγές τους εκεί. Με άλλα λόγια η διαδικασία είναι η παρακάτω για n εργαζομένους:

Πρώτα φτιάχνουμε το φάκελο του project που θα περιέχει και το κύριο *repository*:

```
me@my_pc:~$ mkdir project_master
me@my_pc:~$ cd project_master
me@my_pc:~/project_master$ hg init .
```

Κατόπιν ο κάθε εργαζόμενος έστω i θα πάρει από ένα κλώνο με την παρακάτω διαδικασία:

```
me@my_pc:~$ hg clone project_master project_copy_i
me@my_pc:~$ cd project_copy_i
```

Στη συνέχει ο κάθε εργαζόμενος δουλεύει στο δικό του κλώνο και κάνει *commit* τις αλλαγές στο δικό του *repository* (που βρίσκεται στον αντεγραμμένο φάκελο ουσιαστικά).

Αφού λοιπόν μετά ο εργαζόμενος i τελειώσει τη δουλειά του μπορεί να περάσει τις αλλαγές του στο κύριο *repository* απλά πατώντας την ακόλουθη εντολή:

```
me@my_pc:~/project_copy_i$ hg push
```

Αν στη συνέχεια ο εργαζόμενος j θέλει να ενημερώσει τη δουλειά του με ότι έχει γίνει *push* από τους λοιπούς εργαζόμενους στο κύριο *repository* αρκεί να πατήσει:

```
me@my_pc:~/project_copy_j$ hg pull
```

Βέβαια να σημειωθεί ότι η παραπάνω εντολή απλά ενημερώνει το *repository* και για να δει ο εργαζόμενος j τις αλλαγές θα πρέπει να ενημερώσει το φάκελο του:

```
me@my_pc:~/project_copy_j$ hg update
```

Αν όμως ο εργαζόμενος j δουλεύει σε ένα ίδιο αρχείο με άλλους εργαζομένους και κάποιοι από αυτούς έχουν κάνει κάποια αλλαγή σε αυτό τότε θα πρέπει πρώτα να γίνει *συγχώνευση*[4] των αλλαγών του με αυτές των άλλων πατώντας:

```
me@my_pc:~/project_copy_j$ hg merge
```

Επίσης μετά από μία *συγχώνευση* είναι σύνηθες να γίνονται *commit* οι αλλαγές με σχόλιο τη λέξη *merge*.

```
me@my_pc:~/project_copy_j$ hg commit -m "merge"
```

[4]Καμία φορά το merging μπορεί να μη μπορεί να γίνει αυτοματοποιημένα και ο χρήστης να πρέπει να το κάνει χειροκίνητα. Παρόλα αυτά αυτό είναι ένα αρκετά σπάνιο φαινόμενο.

20 Εξαγωγή του *Patch*

Για να εξάγεται ένα *patch* απλά εκτελείται:

```
me@my_pc:~/project_folder$ hg export $tag_or_revision
```

21 Που κρύβεται όλη η 'ιστορία' του project

Το *repository* με όλες τις αλλαγές που κάνετε κατά καιρούς κρύβεται κάτω από τον κρυφό φάκελο *.hg* στο φάκελο που έχετε δημιουργήσει το *repository* σας.

Για να το δείτε σε Linux / UNIX / Mac OS X αρκεί να πατήσετε:

```
me@my_pc:~/project_folder$ ls -a
```

22 Λοιπά χαρακτηριστικά και επιπλέον πληροφορίες

Το *Mercurial* έχει επίσης υποστήριξη και για διαδικτυακά *repositories* στα οποία μπορεί να έχουμε πρόσβαση είτε μέσω *ssh* είτε μέσω *HTTP*. Για παράδειγμα θα μπορούσε κάποιος να πάρει ένα αντίγραφο του *Mercurial repo*:

```
me@my_pc:~$ hg clone http://selenic.com/hg/
```

και μελλοντικά να πάρει τυχόν αλλαγές:

```
me@my_pc:~/project_folder$ hg pull http://selenic.com/hg/
```

ή για παράδειγμα, αν είχε στηθεί ένας *ssh* server έστω με διεύθυνση *server.com* και σαν χρήστης *user* είχαμε πρόσβαση και κατάλληλα δικαιώματα για το project που βρίσκεται στο φάκελο *project_folder*, τότε θα μπορούσαμε να προωθήσουμε τις αλλαγές μας ως εξής:

```
me@my_pc:~/my_clone$ hg push ssh://user@server.com/project_folder/
```

Παρακάτω παρατίθενται κάποια επιπλέον χρήσιμα links:

- Για μια πιο πλήρη προσέγγιση διαβάστε το 'Mercurial: The Definitive Guide' του Bryan O'Sullivan: `http://hgbook.red-bean.com/read/`

- Στη σελίδα `http://mercurial.selenic.com/wiki/ManPages` μπορείτε να βρείτε μία πλήρη συλλογή με τα απαραίτητα εγχειρίδια χρήσης
- Για ιστότοπους που προσφέρουν *Mercurial* hosting απευθυνθείτε εδώ: `http://mercurial.selenic.com/wiki/MercurialHosting`