差出人: Naruaki TOMA <tnal@ie.u-ryukyu.ac.jp>
件名: [Slab34-ec:00326] zsh, Mercurial 補足
日時: 2012年10月4日 17:47:48 JST
宛先: Evolutionary Computing <slab34-ec@ie.u-ryukyu.ac.jp>
返信先: Evolutionary Computing <slab34-ec@ie.u-ryukyu.ac.jp>

---

zshの話はすっ飛ばしてエンコーディングやエディタの指定ぐらいですが、参考まで。

==========

>zsh 補足
> 漢のzsh
> http://journal.mycom.co.jp/column/zsh/index.html

今回は便利な設定をコピーして利用しましたが、
各設定の意味を理解するには自分で設定を読んだり、
上記サイトやドキュメントを読んでください。
便利な機能についてもいろいろ解説されてます。

==========

>Mercurial 補足
>利用前の準備（個別リポジトリの場合）
(1) 環境変数の設定。
~/.zshrc 等のシェル用設定ファイルで
    export HGENCODING=UTF-8
    EDITOR=emacs
を設定。shark 側は設定済みですので、ローカル側にやっておきましょう。
EDITOR は設定しなくても良いですが、デフォルトだと commit 時の
エディタが vi になります。vi好きな人はviを指定しましょう。

(2) hg 設定ファイルの編集。
commit する際のユーザ名を指定しておきましょう。
~/.hgrc に
    [ui]
    username = Naruaki TOMA <tnal@ie.u-ryukyu.ac.jp>
ぐらいを設定。ローカル側でやっときましょう。
これを設定しておくと、commit 時の username が見つからなかったとかいう
warning が出なくなります。

他にも ignore など便利な機能もありますので、調べてみよう。

(3) リモートでリポジトリを作成。
新しくプロジェクトを始める際には専用のリポジトリを作成してから作業を始めましょう。
ここでは shark にログインし、リポジトリを作成します。
リポジトリの場所は、~info3/HG/2012/ 以下にしてください。
 ・個人利用のものは ~info3/HG/2012/e1057xx/
 ・グループ利用のものは ~info3/HG/2012/groupname/
 ・全員共同利用のものは ~info3/HG/2012/share/
とします。

例えば、e105700 というアカウントの人が個人利用のリポジトリを
つくる場合には以下のようにします。
    macbook> ssh info3@shark.nal.ie.u-ryukyu.ac.jp
    shark% cd HG/2012
    shark% hg init e105700
これで shark 側での作業は終了です。
（他に shark 上で作業する必要が無ければ exit しましょう）

(4) ローカルにリポジトリを複製する。
実際に作業するには、ローカル側で前述のリポジトリを複製し、複製した
リポジトリ内で作業することになります。リポジトリを複製するためには
以下のコマンドを実行しましょう。
    macbook> mkdir temp   #作業用ディレクトリを作成
    macbook> cd temp

macbook> hg clone ssh://info3@shark.nal.ie.u-ryukyu.ac.jp//home/info3/HG/2011/e095700/

引数が長ったらしいですが、これも一度zshで実行しておくと、
「hg」や「hg clone」まで入力した状態で Ctrl+P で以前利用した
引数を自動で補完してくれます。

(5) リポジトリ内で作業する。
ファイル生成したら add して、必要な時に commit します。
ただし、衝突の可能性があるため【commit 前に pull で確認】すべし。（後述:マージの例）
macbook> hg add (ファイル名)
＊ファイル名を省略することが可能ですが、その場合には
未登録のファイル全てが対象になりますので、不要な
ファイルまで登録候補になってないか確認してください。
macbook> hg commit
ただし、この時の commit ではメインリポジトリには反映しません
ので、ローカルのみの管理になります。サーバ上にも反映するには
commit してから push します。
macbook> hg push
普段は commit だけをやっておき、共用が必要になるタイミングで
push する使い方になるでしょう。

(6) 必要に応じて古いバージョンに戻す。
macbook> hg log -v
で更新履歴を参照できるので、そこで付けられてるバージョン番号を
見つけ、
macbook> hg update -r (バージョン番号)
として指定したバージョンに戻します。

(extra.) マージの例。
前提：サーバ側に最新状態があり、ローカル側で古いリポジトリで作業をしてしまった。
以下は test を編集してしまった例。
macbook> emacs test.txt
＊commit 前に pull で確認。
macbook> hg pull
＊サーバ側に最新版があるので問題（衝突）あり！　最新版を取ってきてマージして確認しよう。
macbook> hg update
＊衝突のあったファイルが自動的にマージ（新旧混ざったファイルになる）される。
＊手動で衝突問題を解決し、解決した事を通知する必要がある。
macbook> hg resolve --list # 衝突のあったファイル確認
macbook> emacs test.txt # 衝突のあったファイルを手動で解決
＊編集を終えて commit 版に戻せたらそのことを通知する（終了フラグを立てる）。
macbook> hg resolve -m test.txt
macbook> hg resolve --list # 他にも衝突のあったファイルが無いか確認
＊他にも衝突があればそれらを一通り手動解決する。
＊全てを手動解決後、サーバへのアップロードに戻る。

macbook> hg pull # サーバにアップロードする前に、念のために再度更新が無いか確認
macbook> hg commit
macbook> hg push

(extra.) 必要ないファイルを add 対象にしてしまった場合。
macbook> hg status
macbook> hg revert test.txt~
macbook> hg status
＊問題無ければcommit&push

Naruaki Toma
E-mail: tnal@ie.u-ryukyu.ac.jp, Tel: 098-895-8830
http://www.eva.ie.u-ryukyu.ac.jp/~tnal/