Como usar GIT tras no haber seguido el flujo de trabajo idóneo

Posted: mayo 15th, 2011 | **Author:** fpuga | **Filed under:** General | **Tags:** como, desarrollo sofware, git, how to | 1 Comment »

GIT es una herramienta genial cuando eres un desarrollador disciplinado y sigues el flujo de trabajo recomendado:

```
$ git checkout -b nuevaFuncionalidad
... programar
$ git add -u # añadimos automaticamente todos los fi
$ git add fichero # añadimos los no indexados
$ git commit -m "Mensaje del commit"
... más commits
$ git checkout master
$ git pull --rebase
$ git checkout nuevaFuncionalidad
$ git rebase master
... (solucionar posibles conflictos)
$ git checkout master
$ git rebase nuevaFuncionalidad
$ git push
```

Pero si hay algo que me gusta, y aquí se nota que es una herramienta hecha por desarrolladores para desarrolladores, es lo bien que se comporta cuando no eres tan disciplinado o has metido la pata. Vayamos con un ejemplo (casi) real que se inicia con un viaje en tren Coruña – Vigo.

El estado del repositorio al empezar era este:

```
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit
#
# Changes to be committed:
#   (use "git reset HEAD ..." to unstage)
#
#   modified:   config/text_en.properties
#   modified:   config/text_es.properties
#   modified:   config/text_gl.properties
#   new file:   src/es/udc/cartolab/gvsig/tocextra/Reloa
#
# Untracked files:
#   (use "git add ..." to include in what will be commit
#
#   bin/
```

Se trata de la extensión para gvSIG, extTOCextra desarrollada inicialmente por Javi y

liberada por Cartolab como parte del proyecto gvSIG-EIEL. Como vemos en algún momento del pasado se hizo un commit directamente sobre master que no se subió al repositorio y tenemos cuatro ficheros preparados para hacer un commit que todavía no se ha hecho.

En el viaje de vuelta Vigo – Coruña, nuestro desarrollador se pone a acabar lo que había empezado a la ida, hace un git status y se encuentra que el estado del repositorio es este:

```
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit
#
# Changes to be committed:
#   (use "git reset HEAD ..." to unstage)
#
#   modified:   config/text_en.properties
#   modified:   config/text_es.properties
#   modified:   config/text_gl.properties
#   new file:   src/es/udc/cartolab/gvsig/tocextra/Reload
#
# Changed but not updated:
#   (use "git add ..." to update what will be committed
#   (use "git checkout -- ..." to discard changes in wor
#
```

A base de memoria y git diff sabemos que:

- El commit ya realizado y los archivos preparados para hacer commit constituyen una nueva funcionalidad (funcionalidad 1) que no nos interesa subir al repo por ahora
- La modificación del .classpath y parte de lo modificado en TocExtraExtension son una pequeña refactorización que tiene sentido en si misma y que nos interesa subir como un commit separado del resto
- El nuevo paqueta es.udc.cartolab.gvsig.tocextra.preferences, y los cambios ShowActivesTocMenuEntry, text_es.properties, y parte de los de TocExtraExtension son parte de una nueva funcionalidad (funcionalidad 2) que se empezó a programar y que todavía no está terminada. Así que nos interesa tenerla en una rama disinta de master para poder seguir trabajando sobre ella.

A la vista de esto nuestro objetivo será:

- Tener una nueva rama con la funcionalidad 1 conservando la diferencia entre el commit ya realizado y el que tenemos preparado.
- Una rama nueva con la refactorización en un sólo commit para luego traérnosla a master (tras haber sincronizado master con el repo) y subirla
- Una rama nueva con la funcionalidad 2 en tantos commits como queramos para seguir trabajando.

Para conseguirlo tenemos muchos caminos alternativos, provemos uno en el que usemos distintos comandos que nos permitan ver la potencialidad de git

usemos distintos comandos que nos permitan ver la potencialidad de git.

Acabamos el commit que tenemos preparado

```
$ git commit -m "i18n for ReloadVectorialDBLayerToc
```

Ocultamos temporalmente los cambios que nos quedan para que no nos molesten, creamos una nueva rama para la funcionalidad 1, y limpiamos el master.

```
$ git status
# On branch master
# Your branch is ahead of 'origin/master' by 2 commit
#
# Changed but not updated:
#   (use "git add ..." to update what will be committed
#   (use "git checkout -- ..." to discard changes in wor
#
#   modified:   .classpath
#   modified:   config/text_es.properties
#   modified:   src/es/udc/cartolab/gvsig/tocextra/Show
#   modified:   src/es/udc/cartolab/gvsig/tocextra/TocE
#
# Untracked files:
#   (use "git add ..." to include in what will be commit
```

Creamos una nueva rama para la refactorización y deshacemos la ocultación

```
$ git checkout -b refactor
Switched to a new branch 'refactor'
$ git stash apply
Auto-merging .classpath
CONFLICT (content): Merge conflict in .classpath
Auto-merging config/text_es.properties
CONFLICT (content): Merge conflict in config/text_es
$ git st
# On branch refactor
# Changes to be committed:
#   (use "git reset HEAD ..." to unstage)
#
#   modified:   src/es/udc/cartolab/gvsig/tocextra/Show
#   modified:   src/es/udc/cartolab/gvsig/tocextra/TocE
#
```

Ups, el stash apply nos ha provocado un conflicto. ¿Por qué?. Tómate 15 sg para pensarlo y luego sigue leyendo…

En el commit que teniamos sin hacer había modificaciones sobre text_es.properties, cuando lo comiteamos dejamos algunas modificaciones sobre ese archivo que estaban

cuando lo comiteamos dejamos algunas modificaciones sobre ese archivo que estaban basadas en los cambios comiteados. Como la rama “refactor” la creamos a partir de un master limpio, sin esas modificaciones cuando ejecutamos stash apply ese fichero se encuentra en un estado distinto al esperado y se produce un conflicto que no es capaz de resolver automáticamente. Si hubieramos creado la rama refactor a partir de la rama reloadDBLayers el conflicto no se hubiera producido, pero en esa rama hay cambios que no nos interesan por lo que no podemos hacer esto.

Tras la solución del conflicto el estado de repo es este:

```
# On branch refactor
# Changed but not updated:
#   (use "git add ..." to update what will be committed
#   (use "git checkout -- ..." to discard changes in wor
#
#   modified:   .classpath
#   modified:   config/text_es.properties
#   modified:   src/es/udc/cartolab/gvsig/tocextra/Sho
#   modified:   src/es/udc/cartolab/gvsig/tocextra/TocE
#
# Untracked files:
#   (use "git add ..." to include in what will be commit
#
#   bin/
#   src/es/udc/cartolab/gvsig/tocextra/preferences/
# no changes added to commit (use "git add" and/or '
```

Separar los cambios realizados sobre TocExtraExtension.java

Como los cambios de la refactorización para el archivo TocExtraExtension.java están mezclados con los de la funcionalidad 2, usaremos la opción –patch de git add para separarlos separarlos. Esto nos permitirá escoger que cambios (hunks) de un archivo queremos preparar para comitear en lugar de añadir todo el archivo.

```
$ git add -i src/es/udc/cartolab/gvsig/tocextra/TocExtr
$ git status
# On branch refactor
# Changes to be committed:
#   (use "git reset HEAD ..." to unstage)
#
#   modified:   src/es/udc/cartolab/gvsig/tocextra/TocE
#
# Changed but not updated:
#   (use "git add ..." to update what will be committed
#   (use "git checkout -- ..." to discard changes in wor
#
#   modified:   .classpath
#   modified:   config/text_es.properties
#   modified:   src/es/udc/cartolab/gvsig/tocextra/Sho
```

Como se puede ver, hemos hecho commit de sólo una parte de los cambios realizados en TocExtraExtension. Los cambios que quedan son todos los de la funcionalidad 2, así que los comiteamos a una nueva rama

```
$ git checkout -b newFeature
M   config/text_es.properties
M   src/es/udc/cartolab/gvsig/tocextra/ShowActivesTo
M   src/es/udc/cartolab/gvsig/tocextra/TocExtraExtens
Switched to a new branch 'newFeature'
$ git add -u
$ git add src/es/udc/cartolab/gvsig/tocextra/preferenc
$ git status
# On branch newFeature
# Changes to be committed:
#   (use "git reset HEAD ..." to unstage)
#
#   modified:   config/text_es.properties
#   modified:   src/es/udc/cartolab/gvsig/tocextra/Show
#   modified:   src/es/udc/cartolab/gvsig/tocextra/TocE
```

Subimos al repositorio la refactorización

```
$ git checkout master
$ git pull --rebase
$ git checkout refactor
$ git rebase master
$ git checkout master
$ git merge refactor
$ git push
$ git branch -D refactor
$ git checkout newFeature # para seguir trabajando
```

Parece complicado, pero en realidad es más difícil de leer que de escribir una vez coges un poco de costumbre.