PANDORAFMS

Guidelines:SVN_es

18-11-2021

Guidelines:SVN_es

Repositorio de código SubVersion (SVN)

Un repositorio de código es un sistema que permite trabajar de forma simultánea a varias personas sobre los mismos archivos. Es un sistema que permite además de el trabajo conjunto, llevar un seguimiento, o historial de cada cambio en cada archivo, por cada usuario. Como os podeis imaginar es muy potente. Existe un manual fantástico **y libre** que podeis encontrar en http://svnbook.red-bean.com/.

Como gestor gráfico cliente de Subversion, os recomiendo **esvn** para GNU/Linux.

Instalación de Subversion

Instalate subversion con

```
apt-get install subversion
```

Lea también las normas de estilo de programación

Antes de pensar en subir cambios es importante que hayais leido primero la seccion de "Documentando los cambios" que regulan el formato que deben seguir los comentarios que DEBEIS poner a cada cambio que hagais en el repositorio de código. Podeis leerlo en Guia de estilo:Desarrollo. Documentando los cambios.

Obteniendo una copia del respositorio. Checkout

Lo primero es realizar un "checkout" para descargar el repositorio en tu ordenador y crear de esta manera una copia "local" del repositorio en tu disco duro. Es en esta copia local con la que trabajarás.

Por ejemplo para bajar el repositorio de babel de Sourceforge, valdria el comando:

```
svn checkout https://svn.sourceforge.net/svnroot/babel
```

Subiendo tus cambios al repositorio. Commit

Generalmente los cambios al repositorio sólo los puede subir un desarrollador “estable” del proyecto o el jefe de proyecto. Si se realizan contribuciones esporádicas, lo normal es que se comunique un “parche” en formato DIFF o como extracto de texto a uno de los desarroladores o al jefe de proyecto, de forma que este se encargue de subirlo al repositorio.

Si se tienen privilegios de escritura en el repositorio de Subversion, se pueden subir los cambios automáticamente con 'svn commit'. Cada vez que hayas hecho una modificación y quieras “subirla” al respositorio, tendra que hacer un “commit”. Para ello simplemente tienes que posicionarte en el directorio principal o en el directorio donde cuelgan los ficheros que has modificado y escribir:

```
svn commit
```

Para poder escribir en el SVN hay que tener permisos de escritura y notificar a svn nuestras credenciales. Estas credenciales, una vez usadas quedan almacenadas en la copia local del repositorio (solo tendras que introducirlas la primera vez). Si usas esvn podrás configurarlo en la aplicación. Por linea de comandos tienes que añadir los siguientes parámetros:

```
--username PAR           : especifica un nombre de usuario PAR
--password PAR           : especifica una clave PAR
```

Subversion, al igual que otros sistemas gestores de versiones, permite indicar un resumen con los cambios hechos en ese commit. Para dicho resumen usamos el mismo formato que el ChangeLog, es decir, que copiamos y pegamos los cambios añadidos en este fichero en el resumen del commit.

De esta manera, los ficheros del repositorio mantienen un registro de los cambios que han sufrido, a partir del fichero ChangeLog.

Se te abrirá tu editor por defecto para que introduzcas el mensaje descriptivo del commit que estas realizando. Por favor **cíñete** a las recomendaciones sobre cómo escribir dichos cambios. Un muy buen consejo es: **No hagas commits de cambios pequeños, de forma contínua. Espera a hacer unos cuantos cambios antes de hacer un commit** .

Actualizando tu copia del repositorio. Update

Puede que tengas una copia del repositorio algo antigua y que haya otros desarrolladores que hayan subido cambios desde que actualizaste o te descargarste la copia del repositorio. Para “actualizar” tu repositorio local con los cambios que haya en el

repositorio central, y “mezclar” tu copia local con los cambios del rep. central, tienes que usarl el comando update. Automáticamente te descargará los cambios y te indicará que fichero tiene cambios.

```
svn update
```

Cuando haces un update, te dice además que tipo de actualización se esta produciendo en cada fichero, veamos un ejemplo comentado:

```
C    trunk/pandora_console/include/config.php
U    trunk/pandora_console/help/ca/chap4.php
U    trunk/pandora_console/help/ca/chap5.php
G    trunk/pandora_console/operation/agentes/estado_grupo.php
A    trunk/pandora_agents/win32/configure.in
D    trunk/pandora_agents/win32/pandora_strutils.cpp
```

A la izquierda está el “flag” que indica el tipo de actualización, segun este ejemplo:

- **C** indica conflicto y es lo peor que puede ocurrir, veremos más adelante como solucionar los conflictos. Un conflicto ocurre cuando tu copia local y la copia remota son diferentes y SVN es incapaz de “fusionar” los cambios.
- **U** indica actualización. Es lo más común. Significa que tu copia local estaba desfasada y se ha descargado y aplicado la actualización para ese archivo.
- **A** indica que ha añadido al repositorio ese fichero.
- **D** indica que ha borrado del repositorio ese fichero.
- **G** indica que han producido cambios en un fichero del que tu habias hecho modificación local en tu copia local (y que todavia no habias subido al repositorio), pero que ha podido “fusionar” con éxito a los cambios que habia en el repositorio.

Añadiendo y borrando archivos del repositorio. Add / Delete

Puedes añadir ficheros al repositorio central desde tu repositorio local, simplemente creando un nuevo fichero dentro de la estructura local del repositorio de tu disco duro con el comando add

```
svn add mifichero
```

Puedes borrar un fichero existente en el repositorio central con el comando

```
svn delete fichero
```

Puedes mover un fichero existente en el repositorio central con el comando

```
svn move fichero_antiguo fichero_nuevo
```

No te olvides de documentar los cambios !.

Solucionando conflictos

Como hemos visto, cuando se produce un conflicto, es por que un fichero que habias modificado localmente, tiene cambios en el repositorio central y es imposible “mezclar” esos cambios. La copia central dice A y tu dices B ¿quien tiene razón?. La unica forma de solucionarlo es ver el caso concreto y solucionarlo manualmente.

Pongamos como ejemplo que se ha producido un conflicto en:

```
C    trunk/pandora_console/operation/users/user_statistics.php
```

Si vamos al directorio trunk/pandora_console/operation/users/ veremos que hay cuatro copias del fichero user_statistics.php:

```
user_statistics.php
user_statistics.php.mine
user_statistics.php.r133
user_statistics.php.r151
```

- 'user_statistics.php.r133' Contiene la copia del fichero 'user_statistics.php' tal y como era en la version R133, que es la que tenias en tu repositorio local antes de hacer los cambios, osea, la version original que descargaste a tu copia local antes de hacer los cambios que han generado conflicto.
- 'user_statistics.php.r151' Contiene la copia del fichero 'user_statistics.php' tal y como esta actualmente en el positorio central. Es la “ultima version” almacenada en el repositorio.
- 'user_statistics.php.mine' Contiene mi copia local del fichero. Tal como estaba antes de hacer el update.
- 'user_statistics.php' Contiene la mezcla de los dos cambios con formato DIFF. Es un poco infernal de leer, pero basicamente agrega unos tags, similares a este:

```
<<<<<<< .mine
```

Indicando que lo que hay debajo de esa linea es lo que habia en el fichero .mine, hasta donde pone

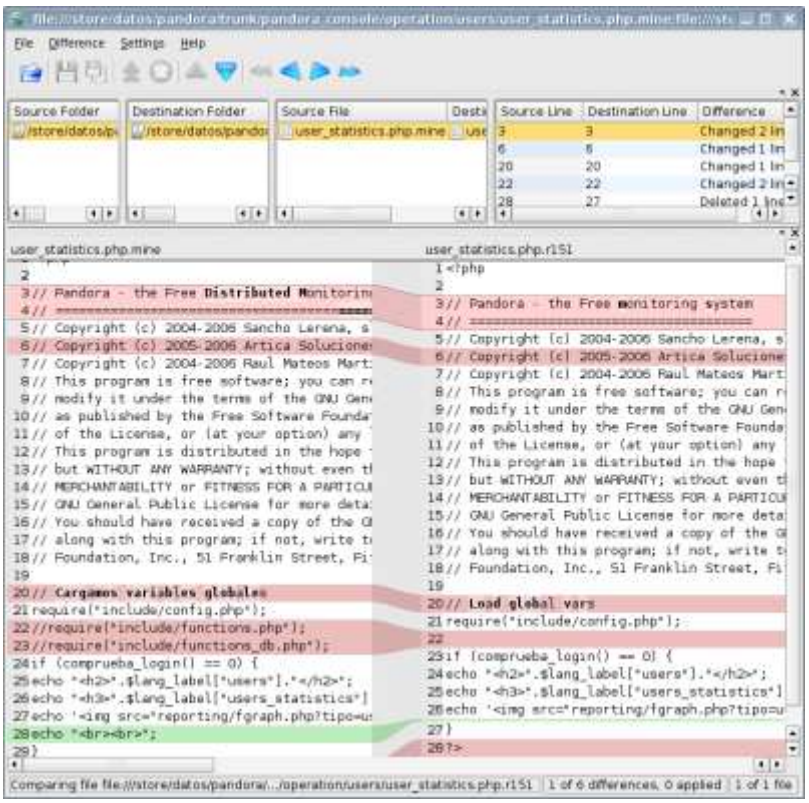
```
=======
```

Y desde esa marca (=====😃 hasta el tag

```
>>>>>>> .r151
```

Son los cambios que hay entre mi version local (.mine) y la version descargada, en este caso la r151 (ver mas abajo sobre historial de versiones).

Otra forma **mucho** mas cómoda de ver las diferencias es usar un gestor gráfico de Diffs, para ello recomiendo el uso de **kompare**. Una captura de esta herramienta vale más que mil palabras:



A Kompare se la pasan dos parámetros: los dos archivos a comparar. Pone en el lazo izquierdo el primero, y en el lado derecho el segundo y permite visualizar las diferencias. Es muy intuitivo y útil.

Una vez que hayas encontrado el problema, edita la ultima versión del fichero en cuestion (en este caso user_statistics.php.r151). Cuando hayas solucionado los cambios, borra localmente (con el comando rm) el fichero “user_statistics.php.mine”, el fichero “user_statistics.php” y el fichero con la copia anterior a tu modificación local, el fichero “user_statistics.php.r133”. De esta manera dejas unicamente el fichero user_statistics.php.r151, que ahora tiene los datos tal y como deberian estar, es decir, con el conflicto “solucionado”. Ahora renombralo al nombre que debe tener “user_statistics.php” y haz un

```
svn commit
```

Es _muy importante_ que comentes que resolviste un conflicto en el log de cambios y porqué se dió ese conflicto, sobre todo si se trata de un conflicto con el trabajo de otro usuario del repositorio.

Versiones

Una de las ventajas mas importantes de SVN es que puedes ver un histórico de cambios por version y autor.

Ver historial de versiones

Puedes ver el historial por cada fichero o por cada directorio o del total. Para ello basta con ir a un directorio de tu copia local del repositorio y escribir :

```
svn log
```

Esto te mostrará el contenido de lo que se ha hecho en cada revision (por eso la R delante del numero de revision), lo que visualiza es básicamente lo que el usuario ha subido como comentario al subir los cambios, por eso es tan importante seguir una norma establecida a la hora de documentar los cambios. Veamos un ejemplo:

```
------------------------------------------------------------------------
r151 | esanchezm | 2006-08-31 14:42:36 +0200 (jue, 31 ago 2006) | 8
lines

2006-08-31  Esteban Sanchez  <estebans@artica.es>

        * pandora.cc, pandora_strutils.h, windows_service.h,
wmi/pandora_wmi.cc, ssh/pandora_ssh_test.cc:  Documentation
comments updated.

        * ssh/pandora_ssh_client.[h,cc]: Documentation comments
updated. Comment deleted.

        * windows_service.cc: Comment deleted.
------------------------------------------------------------------------
r150 | esanchezm | 2006-08-25 15:02:03 +0200 (vie, 25 ago 2006) |
46 lines

2006-08-15  Esteban Sanchez  <estebans@artica.es>
```

```
        * autogen.sh, configure.in, Makefile.am, Doxyfile.in: Added
to
        repository. They are used to generate documentation, not to
compile.

        * main.cc: Added some comments. Style correction.

        * pandora.[cc,h]: Added documentation comments. Changed
visibility of
        some attributes.
```

Volver a una versión anterior

Para descargar una version concreta del repositorio, esto es, para "volver atrás en el tiempo", sólo tienes que llamar a svn update con el número de revisión que quieres, por ejemplo, imaginemos que visto el log anterior queremos volver a la revision R150 del fichero *autogen.sh*. Bastaria el comando

```
svn update -r 150 win32/autogen.sh
```

Ver el contenido de una version anterior

Si sólo quieramos echar un vistazo, sin alterar nuestra copia local, podriamos usar el comando **cat**

```
svn cat -r 150 win32/autogen.sh
```

Ver diferencias entre dos versiones del repositorio

Muestra en formato DIFF las diferencias entre dos versiones de un fichero. En este ejemplo se usa la palabra espcial "HEAD" que hace referencia a la ultima version del repositorio. De forma que este ejemplo muestra los cambios del fichero *win32/pandora_windows_service.h* entre la versión 130 y la versión actual del repositorio:

```
svn diff -r 130:HEAD win32/pandora_windows_service.h
```

Branches / Ramas

Los branches o ramas, son versiones "diferentes" de la version principal, que esta a partir del directorio "trunk". Para cada version se generará una rama propia, que mantendra los

cambios locales para esa version. Cuando haya que “propagar” un cambio se hara un *merge* de una rama al trunk o viceversa. Estos *merges* seran siempre coordinados con el jefe de proyecto, o directamente ejecutados por el jefe de proyecto.

Si un desarrollador esta implementando un subproyecto o una parte importante de codigo que requiere de muchos cambios sobre la version de desarrollo, es una opcion recomendable crear un branch local para trabajar con ella y posteriormente hacer un merge con la version de desarrollo principal. La creacion de ramas locales debe ser coordinada con el Jefe de proyecto para una mejor coordinacion del proyecto.