FACULTAD DE MATEMÁTICAS

Tutorial de Subversion

Uso del subversion desde el Eclipse y con el cliente TortoiseSVN

27/08/2009

Este tutorial tiene como objetivo sensibilizar a los desarrolladores del proyecto “Scensy v2.0” en el uso de la herramienta Subversion para la administración de la documentación y código fuente generado en el mismo.

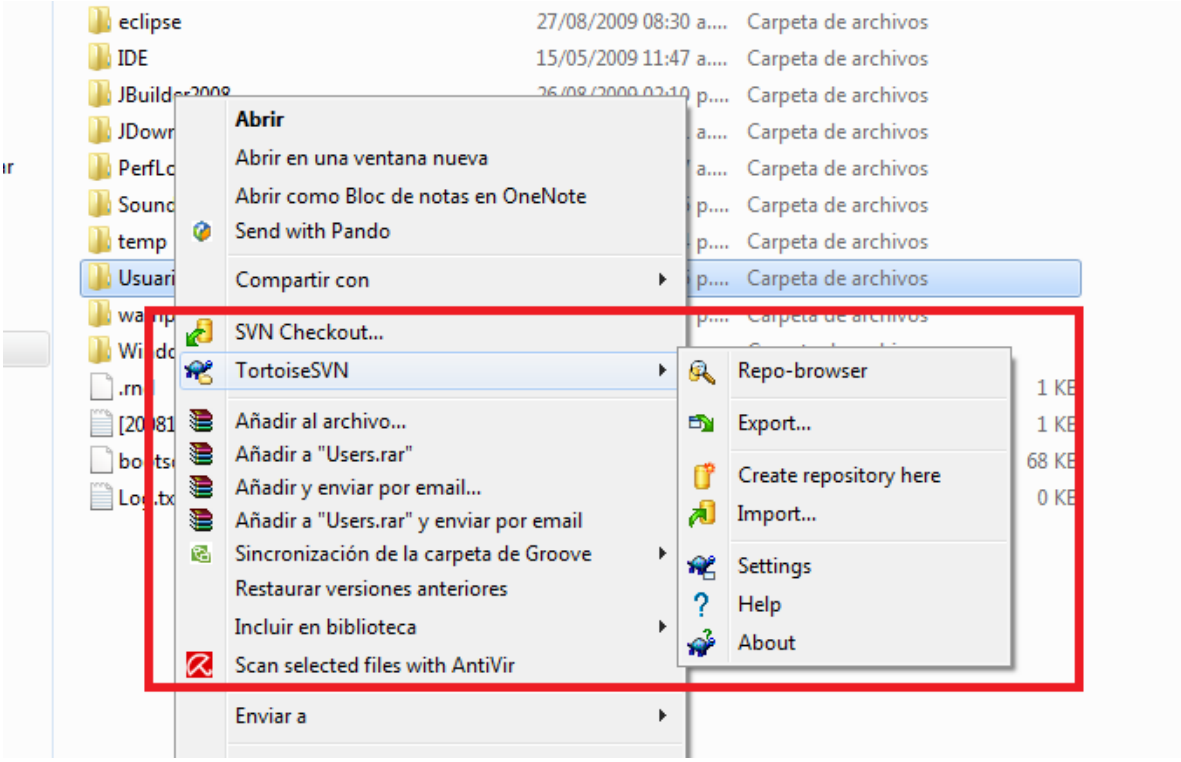
Contenido

Requerimientos

1. Eclipse
2. Instalación del SVN Tortoise
3. Instalación Subversive en Eclipse

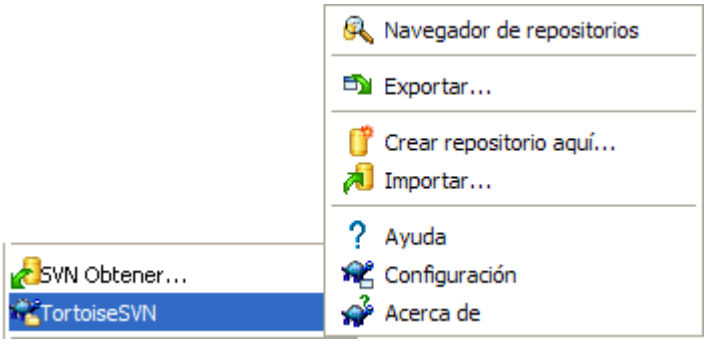
Uso del TortoiseSVN

La herramienta TortoiseSVN se integra dentro del explorador de Windows. Para usar TortoiseSVN es necesario abrir el explorador de Windows y hacer clic con el botón derecho del mouse en cualquier carpeta que desees ejecutar algún comando del TortoiseSVN.



Creando el repositorio con TortoiseSVN

El menú de TortoiseSVN para carpetas no versionadas

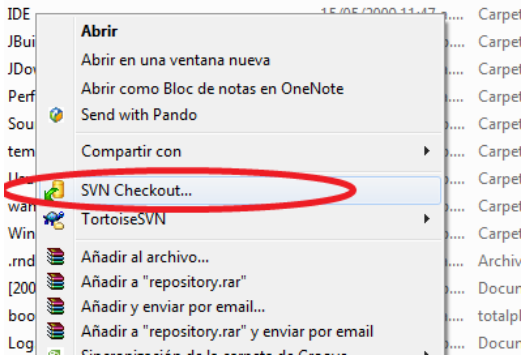


1. Abra el explorador de Windows
2. Cree una nueva carpeta y llámela por ejemplo `scensy_repository`
3. Haga clic con el botón derecho sobre la carpeta recién creada y seleccione TortoiseSVN → Crear Repositorio aquí....

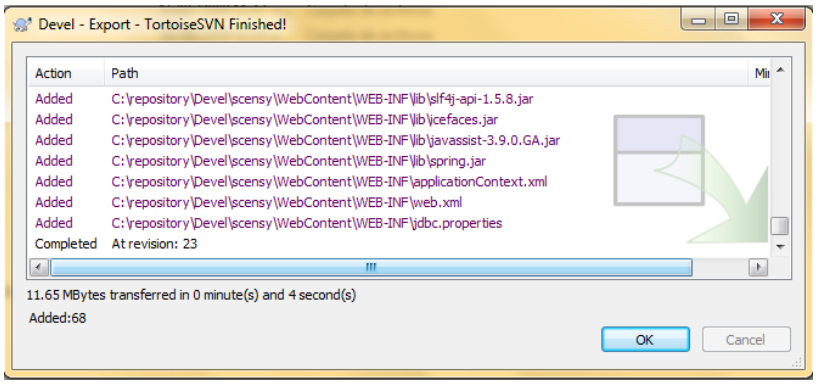
Accediendo a un repositorio en una unidad de red

Para utilizar TortoiseSVN (o cualquier otro cliente de Subversion), necesita un lugar donde poner sus repositorios. Puede o bien almacenar sus repositorios de forma local y acceder a ellos utilizando el protocolo `file://`, o puede ponerlos en un servidor y acceder a ellos con los protocolos `http://` o `svn://`. Los dos protocolos de servidor también pueden estar encriptados. Utilice `https://` o `svn+ssh://`, o puede utilizar `svn://` con SASL.

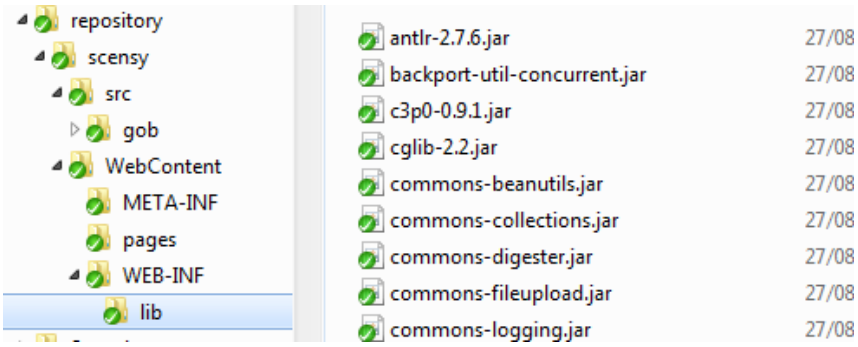
En el ambiente de desarrollo de "Scensy", el repositorio estará almacenado remotamente, por lo que se utilizará una ruta como ejemplo: *svv://192.168.57.152/Devel*, para ello es necesario hacer clic en la opción "SVN Checkout…"



Luego de ello capturar la ruta anterior en el campo de texto y hacer clic en el botón con la etiqueta "OK" y en ese momento iniciara la descarga de la estructura de directorios y archivos existentes en el repositorio remoto con el status correspondiente.



La estructura de directorios generados tendrá la siguiente estructura:



Una vez creada la estructura se puede trabajar como se acostumbra, creando documentos y modificándolos (en este caso cambiará el status del archivo a modificado para posteriormente hacerle un commit al repositorio remoto), de tal manera que al final de sus cambios pueda subirlos en el repositorio remoto.

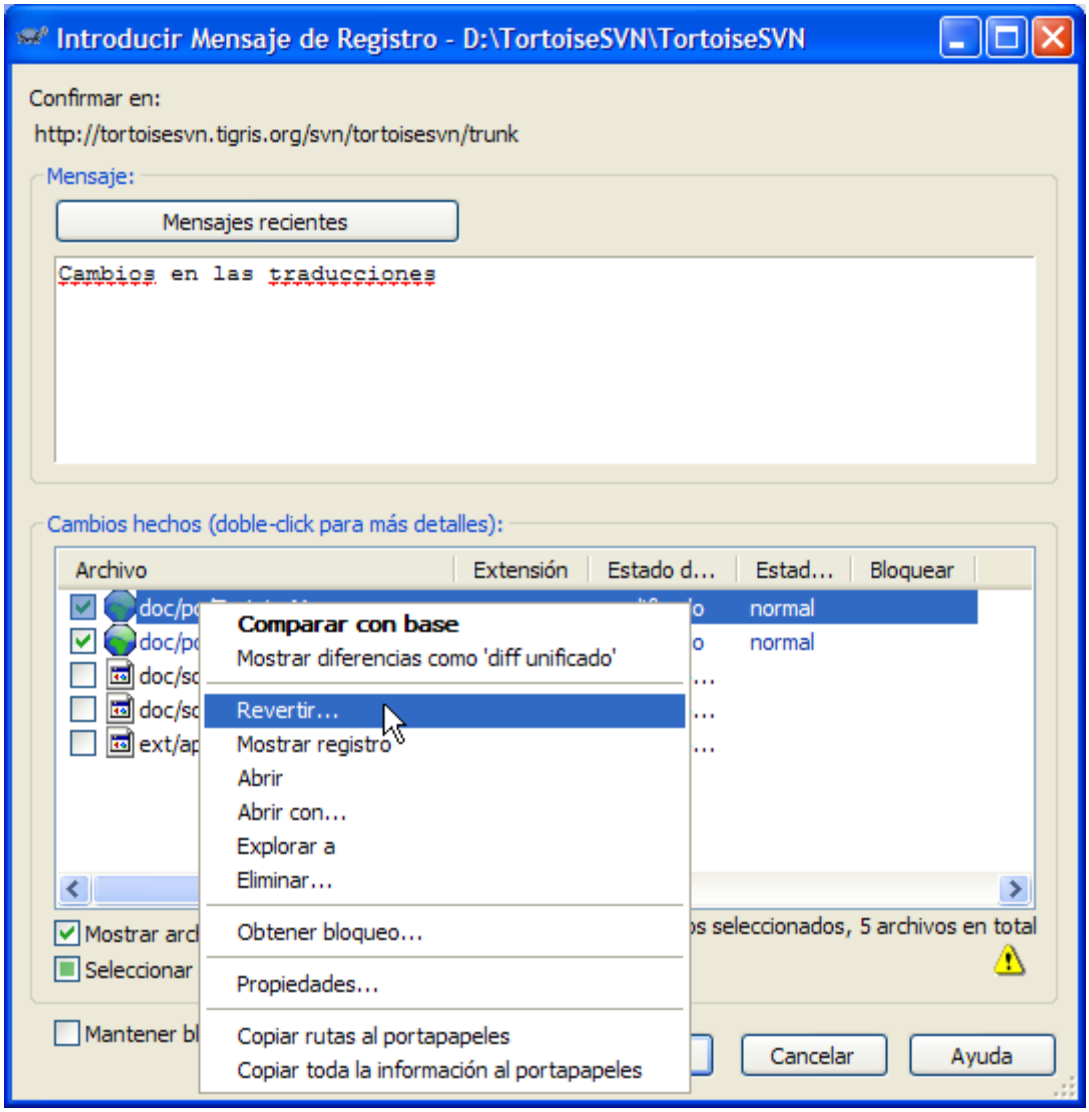
Confirmando sus cambios en el repositorio

Enviar los cambios que ha hecho al repositorio se conoce como *confirmar* los cambios. Pero antes de confirmar tiene que estar seguro de que su copia de trabajo está actualizada. Puede o bien ejecutar "SVN Update", o bien ejecutar TortoiseSVN → "Check for modifications" primero, para ver qué se ha cambiado localmente o en el servidor.

"commit"

Si su copia de trabajo está actualizada y no hay conflictos, ya está preparado para confirmar sus cambios. Seleccione los ficheros y/o carpetas que desee confirmar y seleccione "SVN commit"

El diálogo de Confirmación



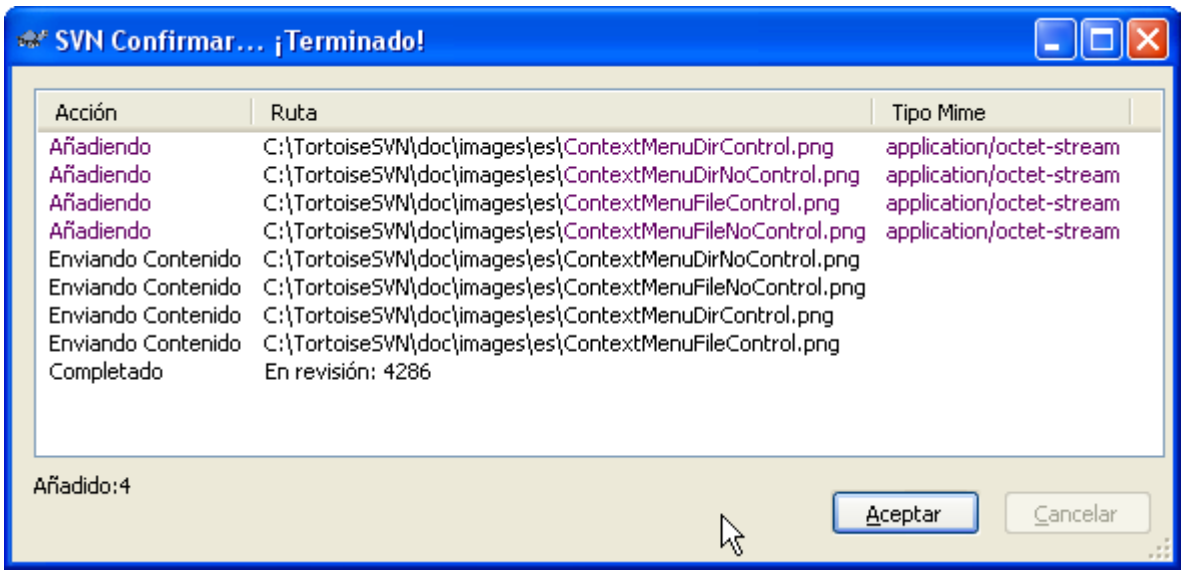
El diálogo de confirmación le mostrará todos los ficheros cambiados, incluso los ficheros añadidos, borrados o no versionados. Si no desea que un fichero cambiado se confirme, simplemente desmarque ese fichero. Si desea incluir un fichero no versionado, márquelo para añadirlo a la confirmación.

Los ítems que han sido cambiados a una ruta de repositorio diferente también se indican utilizando un marcador `(s)`. Puede haber cambiado algo mientras trabaja en una rama y habérsele olvidado volver a cambiarlo al tronco. ¡Este es su signo de advertencia!

Progreso de confirmación

Tras pulsar OK aparece un diálogo mostrando el progreso de la confirmación.

Figura 4.10. El diálogo Progreso mostrando el progreso de una confirmación



El diálogo de progreso utiliza una codificación de colores para resaltar las diferentes acciones de confirmación:

Azul

: Confirmando una modificación.

Púrpura

: Confirmando un ítem añadido.

Rojo oscuro

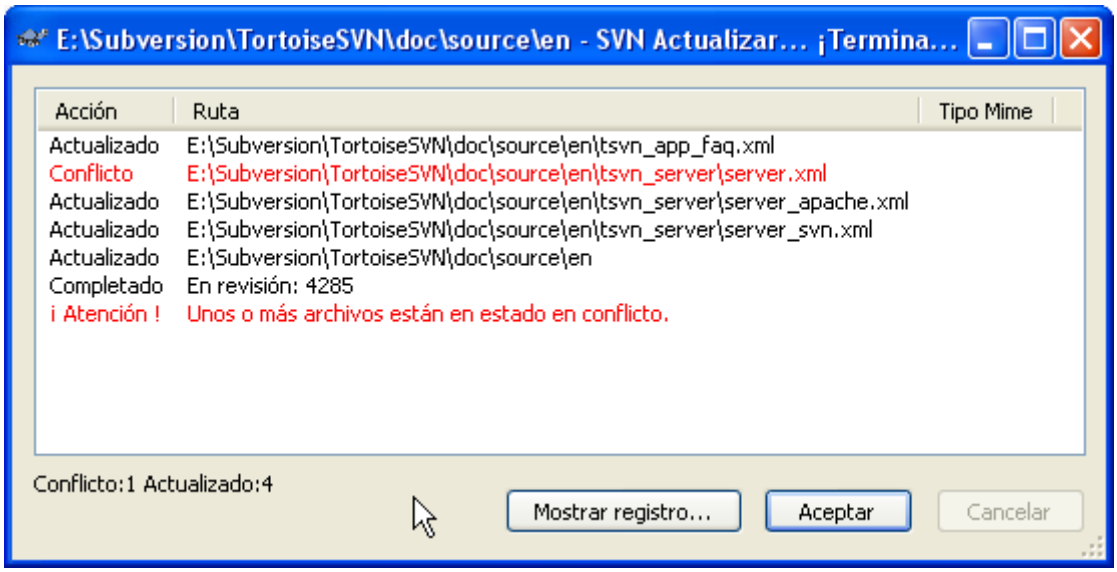
: Confirmando un borrado o un reemplazo.

Negro

: Todos los demás ítems.

Actualice su copia de trabajo con los cambios de otros

Diálogo de progreso mostrando una actualización terminada



Periódicamente, debería asegurarse de que los cambios que hacen los demás se incorporen en su copia de trabajo local. El proceso de incorporar los cambios desde el servidor a su copia de trabajo local se conoce como *"Update"*. La actualización puede hacerse en ficheros sueltos, en un conjunto de ficheros, o recursivamente en jerarquías completas de directorios. Para actualizar, seleccione los ficheros y/o directorios que desee, haga clic con el botón derecho y seleccione "SVN Update" en el menú contextual del explorador. Aparecerá una ventana con el progreso de la actualización según se ejecuta. Los cambios que los demás hayan hecho se fusionarán con sus ficheros, manteniendo cualquier cambio que haya hecho en los mismos ficheros. El repositorio *no* se ve afectado por una actualización.

El diálogo de progreso utiliza un código de colores para resaltar diferentes acciones de actualización:

Púrpura

: Nuevo ítem añadido a su copia de trabajo

Rojo oscuro

: Ítem redundante borrado de su copia de trabajo, o ítem faltante reemplazado en su copia de trabajo.

Verde

Cambios del repositorio que se han fusionado satisfactoriamente con sus cambios locales.

Rojo brillante

Cambios del repositorio fusionados con sus cambios locales, pero que han dado lugar a conflictos que debe resolver.

Negro

Ítems sin cambios en su copia de trabajo actualizados con una versión más nueva desde el repositorio.

La opción "*Update to version*" puede ser útil a veces para ver cómo estaba su proyecto en un momento anterior en su historia. Pero en general, actualizar ficheros individuales a una revisión anterior no es una buena idea, ya que deja su copia de trabajo en un estado inconsistente. Si el fichero que está actualizando ha cambiado de nombre, incluso puede encontrar que ese fichero ha desaparecido de su copia de trabajo porque en esa revisión no había ningún fichero con ese nombre. También debe tener en cuenta que el ítem mostrará una sobreimpresión normal verde, por lo que no se puede distinguir del resto de ficheros que están actualizados.

Resolviendo conflictos

De vez en cuando, obtendrá un *conflicto* cuando actualice/fusione sus ficheros desde el repositorio o cuando cambie su copia de trabajo a una URL diferente. Hay dos tipos de conflictos:

Conflictos de fichero

Un conflicto de fichero ocurre si dos (o más) desarrolladores han cambiado las mismas líneas de un fichero.

Conflictos de árboles

Un conflicto de árbol ocurre cuando un desarrollador mueve/renombra/elimina un fichero o una carpeta, que otro desarrollador también ha movido/renombrado/borrado, o quizás lo haya modificado.

Conflictos de ficheros

Un conflicto de fichero ocurre cuando uno o más desarrolladores han hecho cambios en las mismas líneas de un fichero. Dado que Subversion no sabe nada de su proyecto, delega la resolución de los conflictos en los desarrolladores. Cuando se le informa de un conflicto, debería abrir el fichero en cuestión, y buscar líneas que empiecen con el texto `<<<<<<<`. El área conflictiva se marca así:

```
<<<<<<< nombre-del-fichero
        sus cambios
=======
        código fusionado del repositorio
>>>>>>> revisión
```

Además, para cada fichero en conflicto Subversion deja tres ficheros adicionales en su directorio:

nombre-del-fichero.ext.mine

> Este es su fichero tal y como estaba en su copia de trabajo antes de que actualizara su copia de trabajo - esto es, sin marcadores de conflicto. Este fichero tiene sus últimos cambios en él y nada más.

nombre-del-fichero.ext.rREV-ANTIGUA

> Este es el fichero que era la revisión BASE antes de que actualizara su copia de trabajo. Esto es, el fichero que obtuvo antes de empezar a hacer sus últimos cambios.

nombre-del-fichero.ext.rREV-NUEVA

> Este es el fichero que su cliente de Subversion acaba de recibir desde el servidor del que actualizó su copia de trabajo. Este fichero corresponde a la revisión HEAD del repositorio.

Puede o bien lanzar una herramienta externa de fusiones / editor de conflictos con el menú contextual TortoiseSVN → Editar Conflictos o bien utilizar otro editor manualmente para resolver el conflicto. Debe decidir cómo tiene que quedar el código, hacer los cambios necesarios, y grabar el fichero.

Después, ejecute el comando TortoiseSVN → Resolver y confirme sus modificaciones al repositorio. Tome nota de que el comando Resolver realmente no resuelve el conflicto. Simplemente elimina los ficheros `filename.ext.mine` y `filename.ext.r*`, dejándole confirmar sus cambios.

Si tiene conflictos con ficheros binarios, Subversion no intentará mezclar dichos ficheros por sí mismo. El fichero local se mantendrá sin cambios (exactamente tal y como lo había

cambiado usted) y obtendrá unos ficheros `nombrefichero.ext.r*`. Si desea descartar sus cambios y quedarse con la versión del repositorio, utilice el comando Revertir. Si desea mantener su versión y sobrescribir la versión del repositorio, utilice el comando Resuelto y luego confirme su versión.

Puede utilizar el comando Resuelto para múltiples ficheros si pulsa con el botón derecho en la carpeta padre y selecciona TortoiseSVN → Resuelto... Esto mostrará un diálogo con todos los ficheros en conflicto dentro de esa carpeta, y le permitirá seleccionar cuáles marcar como resueltos.

Conflictos de árbol

Un conflicto de árbol ocurre cuando un desarrollador mueve/renombra/elimina un fichero o una carpeta, que otro desarrollador también ha movido/renombrado/borrado, o quizás lo haya modificado. Hay diferentes situaciones que puede resultar en un conflicto de árbol, y todas ellas requieren pasos diferentes para resolver el conflicto.

Cuando se elimina un fichero de forma local en Subversion, el fichero también se elimina del sistema local de ficheros, por lo que incluso si es parte de un conflicto de árbol no se puede mostrar una sobreimpresión de conflicto y no puede hacer clic con el botón derecho sobre él para resolver el conflicto. En este caso, utilice el diálogo Comprobar modificaciones para acceder a la opción Editar conflictos.

TortoiseSVN puede ayudarle a encontrar el lugar correcto para fusionar los cambios, pero puede que necesite realizar un trabajo adicional para arreglar los conflictos. Recuerde que tras una actualización la BASE de trabajo siempre contendrá la revisión de cada ítem tal y como estaba en el repositorio en el momento de la actualización. Si revierte un cambio tras la actualización, se revierte a su estado del repositorio, no a como estaba cuando empezó a hacer sus propios cambios locales.

Borrado local, llega un cambio en la actualización

1. El desarrollador A modifica `Foo.c` y lo confirma en el repositorio
2. El desarrollador B al mismo tiempo ha movido `Foo.c` a `Bar.c` en su propia copia de trabajo, o simplemente ha borrado `Foo.c` o su carpeta padre.

Una actualización de la copia de trabajo del desarrollador B acaba con un conflicto de árbol:

- `Foo.c` ha sido borrado de la copia de trabajo, pero está marcado como un conflicto de árbol.
- Si el conflicto aparece después de un renombrado en vez de un borrado, entonces `Bar.c` está marcado como añadido, pero no contiene las modificaciones del desarrollador A.

El desarrollador B ahora tiene que elegir si mantiene los cambios realizados por el desarrollador A. En el caso de un renombrado, puede fusionar los cambios de `Foo.c` dentro del fichero renombrado `Bar.c`. Para simples borrados de ficheros o directorios puede elegir quedarse con los cambios del ítem del desarrollador A y descartar el borrado. O, si marca el conflicto como resuelto sin hacer nada más, estará descartando los cambios del desarrollador A.

El diálogo de editar conflictos ofrece la posibilidad de fusionar cambios si puede encontrar el fichero original del renombrado `Bar.c`. Dependiendo de dónde se haya realizado la actualización, puede que no sea posible encontrar el fichero de origen.

Edición local, entra un borrado en la actualización

1. El desarrollador A mueve `Foo.c` a `Bar.c` y lo confirma en el repositorio.
2. El desarrollador B modifica `Foo.c` en su copia de trabajo.

O en el caso de mover una carpeta...

1. El desarrollador A mueve la carpeta padre `FooFolder` a `BarFolder` y lo confirma en el repositorio.
2. El desarrollador B modifica `Foo.c` en su copia de trabajo.

Una actualización de la copia de trabajo de B acaba con un conflicto de árbol. Para un conflicto simple de fichero:

- `Bar.c` se añade a la copia de trabajo como un fichero normal.
- `Foo.c` se marca como añadido (con historia) y tiene un conflicto de árbol.

Para un conflicto de carpeta:

- `BarFolder` se añade a la copia de trabajo como una carpeta normal.
- `FooFolder` se marca como añadida (con historia) y tiene un conflicto de árbol.

  `Foo.c` se marca como modificado.

El desarrollador B tiene ahora que decidir si desea continuar con la reorganización del desarrollador A y fusionar sus cambios en los ficheros correspondientes de la nueva estructura, o simplemente revertir los cambios de A y mantener el fichero local.

Para fusionar sus cambios locales con la reorganización, el desarrollador B tiene que encontrar primero qué nombre de fichero tiene ahora el fichero en conflicto `Foo.c` que fue renombrado/movido en el repositorio. Esto puede hacerse utilizando el diálogo de registro. Luego debe fusionar los cambios a mano ya que no hay actualmente forma de automatizar o simplificar este proceso. Una vez que se hayan portado los cambios, la ruta en conflicto

es redundante y puede borrarse. En este caso utilice el botón Eliminar en el diálogo de editar conflictos para limpiar y marcar el conflicto como resuelto.

Si el desarrollador B decide que los cambios de A eran erróneos deberá elegir el botón Mantener en el diálogo de editar conflictos. Esto marca el fichero o carpeta en conflicto como resuelto, pero los cambios del desarrollador A tendrán que eliminarse a mano. De nuevo el diálogo de registro ayuda a controlar lo que se ha movido.

Eliminación local, entra una eliminación en la actualización

1. El desarrollador A mueve `Foo.c` a `Bar.c` y lo confirma en el repositorio
2. El desarrollador B mueve `Foo.c` a `Bix.c`

Una actualización de la copia de trabajo del desarrollador B acaba con un conflicto de árbol:

- `Bix.c` se marca como añadido con historia.
- `Bar.c` se añade a la copia de trabajo con el estado 'normal'.
- `Foo.c` se marca como borrado y tiene un conflicto de árbol.

Para resolver este conflicto, el desarrollador B tiene que averiguar qué nombre de fichero tiene ahora el fichero en conflicto `Foo.c` que fue renombrado/movido en el repositorio. Esto puede hacerse utilizando el diálogo de registro.

Luego el desarrollador B tiene que decidir qué nuevo nombre de fichero de `Foo.c` mantiene - el del desarrollador A o el renombrado que hizo él mismo.

Después de que el desarrollador B haya resuelto manualmente el conflicto, el conflicto de árbol debe marcarse como resuelto mediante el botón del diálogo de editar conflictos.

Falta en local, entra un cambio en la fusión

1. El desarrollador A, que está trabajando en el tronco, modifica `Foo.c` y lo confirma en el repositorio
2. El desarrollador B, que está trabajando en una rama, mueve `Foo.c` a `Bar.c` y lo confirma en el repositorio

Una fusión de los cambios en el tronco del desarrollador A con los cambios de la copia de trabajo de la rama del desarrollador B acaba con un conflicto de árbol:

- `Bar.c` ya está en la copia de trabajo con estado 'normal'.
- `Foo.c` se marca como faltante con un conflicto de árbol.

Para resolver este conflicto, el desarrollador B tiene que marcar el fichero como resuelto en el diálogo de edición de conflictos, lo que lo eliminará de la lista de conflictos. Luego tendrá que decidir si copia el fichero faltante `Foo.c` desde el repositorio a la copia de trabajo, si fusiona los cambios del desarrollador A hechos a `Foo.c` en el fichero renombrado `Bar.c`, o si desea ignorar los cambios marcando el conflicto como resuelto y no haciendo nada más.

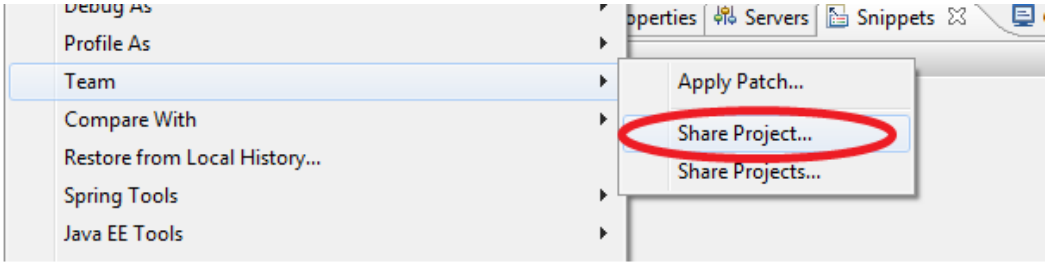
Tenga en cuenta que si copia el fichero faltante desde el repositorio y luego marca como resuelto, su copia de eliminará de nuevo. Tiene que resolver el conflicto antes.

Uso del Subversive en el Eclipse

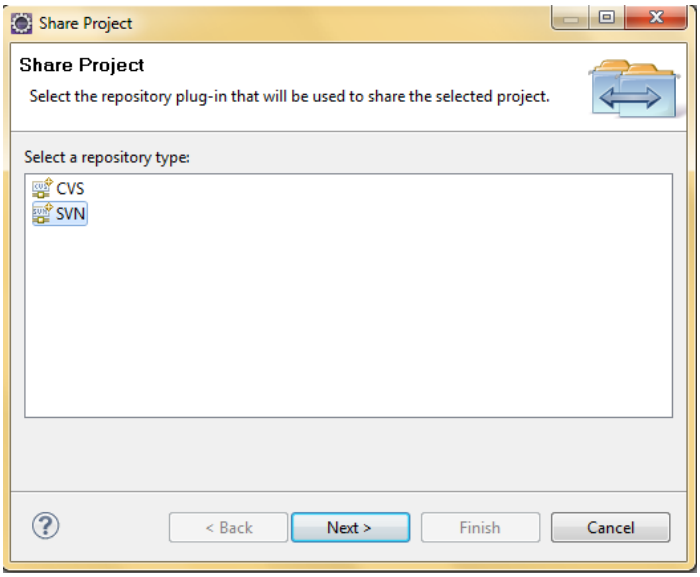
El Subversive es un plugin que se instala en el Eclipse, el cual, nos permite interactuar directamente nuestros proyectos de eclipse con los repositorios locales y remotos que trabajan con Subversion, esto nos permite una mayor facilidad a la hora de programar.

Crear un proyecto con conexión al repositorio a través del Subversive

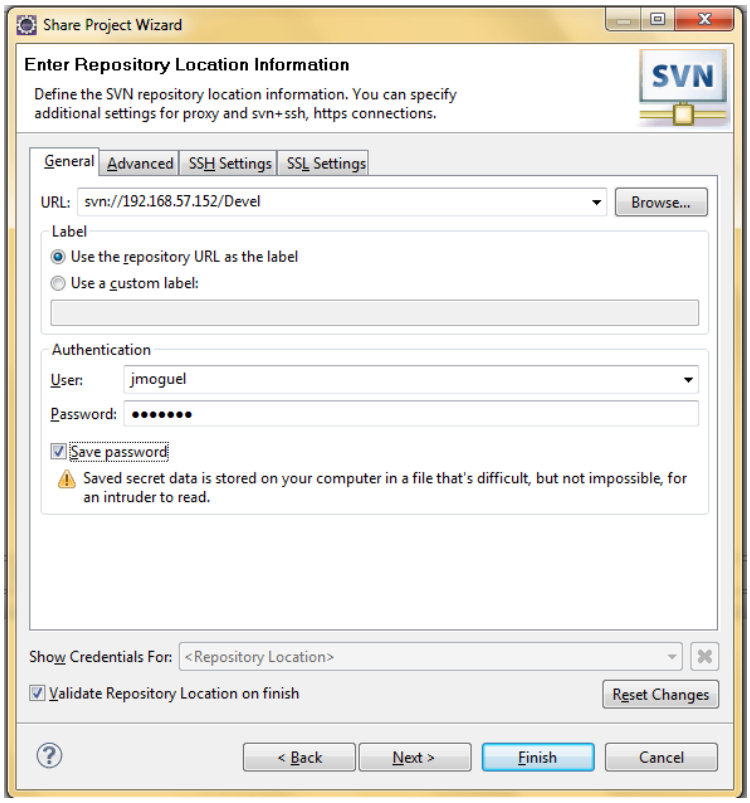
Para tener un proyecto con subversion, es necesario, crear la estructura del proyecto (Proyecto Web, proyecto básico, Web Services, etc.). Una vez que tengamos nuestro proyecto creado con el eclipse, hacemos clic con el botón derecho del mouse sobre el proyecto y seleccionamos la opción ***Team>Share Project***.



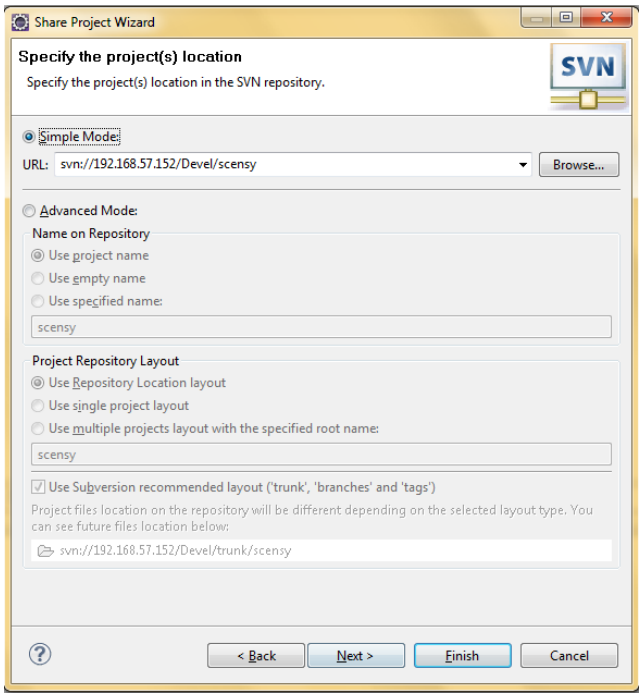
Seleccionar la opción "**SVN**" de la ventana Share Project y seleccione "**Next >**".



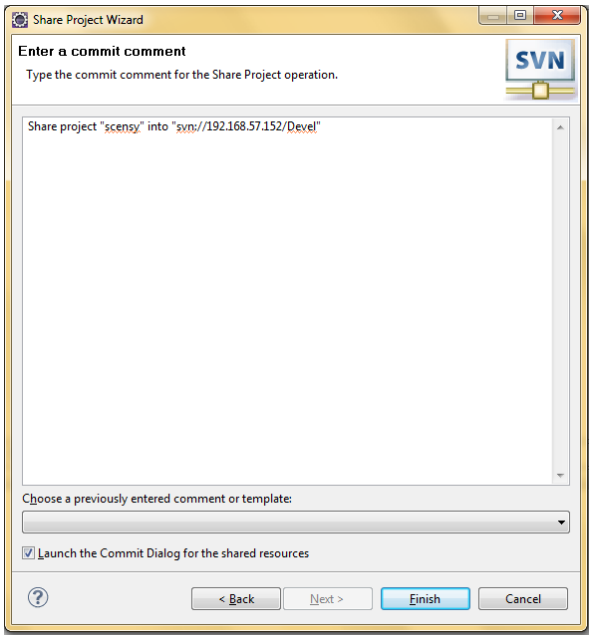
Capturar los valores de conexión y autentificación al repositorio remoto y seleccione "**Next >**":



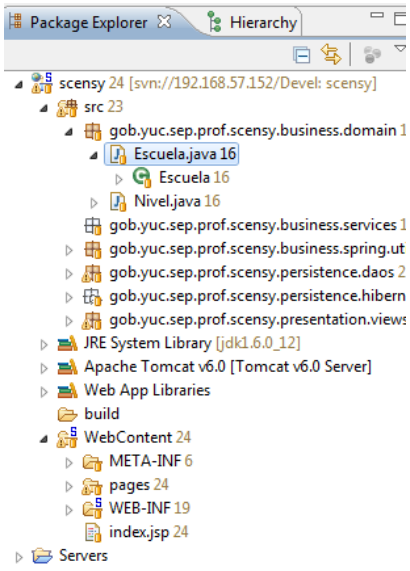
Seleccionar la ruta y el nombre de localización del proyecto en el repositorio remoto y seleccione "**Next >**":



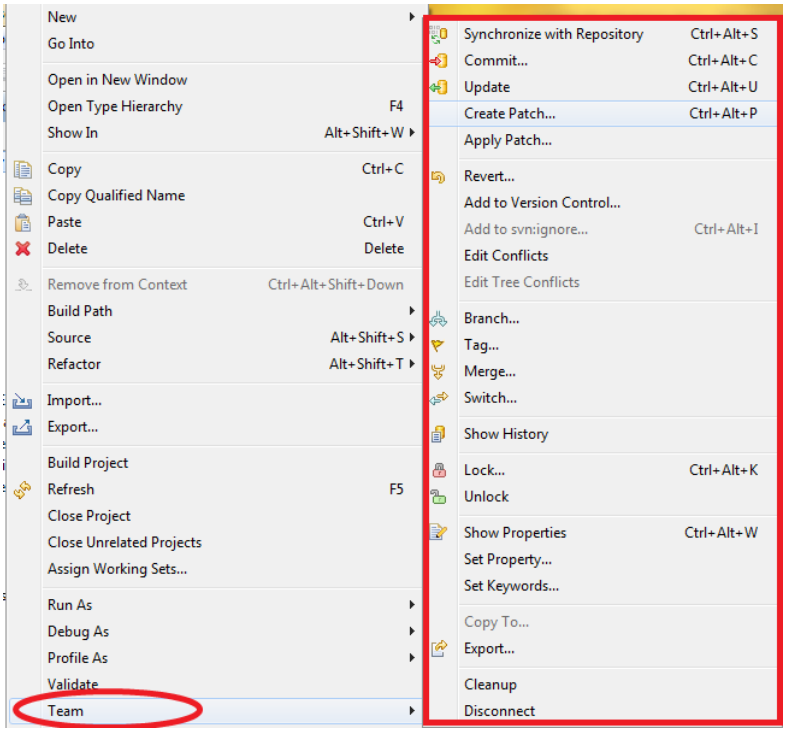
Captura algún comentario que tendrá el commit (En este caso es el commit inicial al repositorio remoto) y haz clic en "**Finish**"



Al finalizar este proceso, se descarga la versión más reciente que hay en el repositorio de Subversion.



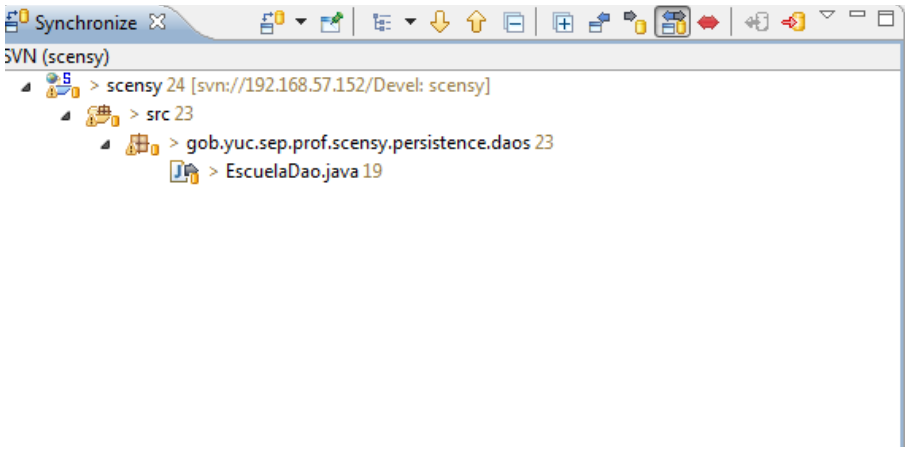
Para ver todas las opciones que tiene el Subversive en el eclipse, es hacer clic derecho del mouse sobre el proyecto y seleccionar la opción "**Team**".



Commit

Durante el desarrollo, es normal que los programadores realicen cambios a los archivos ya existentes o incluso agregar nuevos archivos de código fuente o librerías. Por lo tanto es necesario agregar estos cambios al repositorio del control de versiones. Para esto, es importante sincronizar

nuestro proyecto con el repositorio remoto en busca de inconsistencias (**Team → Sinchronize with Repository**):



En la ventana anterior, se puede ver todos los cambios que ha realizado y que han realizado los demás desarrolladores en el ambiente de desarrollo. Además de encontrar inconsistencias como la modificación de la misma línea de un mismo archivo del código fuente. En esta ventana se puede seleccionar elemento a elemento o seleccionar varios a la vez, a través del botón derecho sobre el o los archivos con el mouse y seleccionar la opción "**commit…**". De la misma forma, de existir cambios que no tienes en tu proyecto puedes seleccionarlos y utilizar la opción "**Update**", la cual actualizará los cambios a tu proyecto local. En el caso de existir inconsistencias, se puede hacer doble clic al archivo y verificar en que línea están las inconsistencias, de tal forma que se pueda resolver de manera manual o seleccionar cualquiera de las opciones **Override and Commint** y **Override and Update.** Las cuales reemplazarán el archivo en el repositorio remoto ó reemplazarán los cambios en tu proyecto local respectivamente.