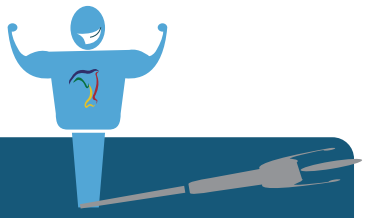
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

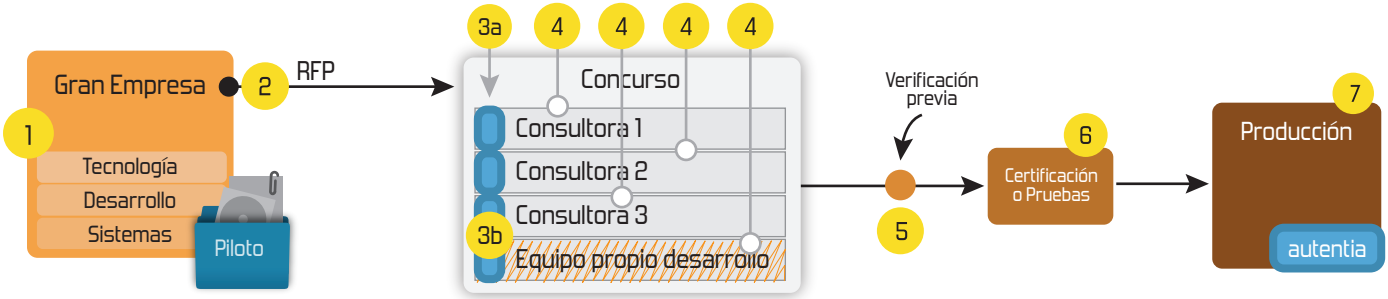
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

Contacte con Cristóbal González
criskerberos-tutoriales@yahoo.com

DESARROLLADO POR:
Cristóbal González Almirón

Consultor de desarrollo de proyectos informáticos. Su experiencia profesional se ha desarrollado en empresas como Compaq, HP, Mapfre, Endesa, Repsol, Universidad Autónoma de Madrid, en las áreas de Desarrollo de Software (Orientado a Objetos), tecnologías de Internet, Técnica de Sistemas de alta disponibilidad y formación a usuarios.

Ultimas Noticias

- Comic Flash de Head Hunting
- XI Charla Autentia - Mule - Recordatorio
- Comparte el Conocimiento en Adictos
- ¡¡¡ Alcanzamos los 900 tutoriales !!!
- Persiguiendo la felicidad, haciendo realidad los sueños

Fecha de publicación del tutorial: 2010-10-01 4

Rendimiento en espacio y transferencia de un servidor Subversion

Resumen

En este artículo estudiaremos el rendimiento de un repositorio Subversion desde el punto de vista del espacio que consumen las diferentes operaciones que realizamos sobre él: inserción de ficheros, copia, creación de ramas y etiquetas, así como la cantidad de información transferida entre el servidor y la copia local en dichas operaciones.

Introducción

En el artículo anterior dedicado a la instalación de un servidor Subversion para Windows y su configuración en modo portable, pudimos comprobar gran sencillez con la que podemos poner a punto nuestro servidor de control de versiones Subversion. Pero nos surge probablemente una pregunta, y es cómo de eficiente es el repositorio Subversion.

Bien, hace unos meses tuve la oportunidad de trabajar con un grupo de auténticos profesionales de la informática de Málaga, a los que desde aquí aprovecho para saludar, que fueron los que me motivaron a realizar este artículo.

Cuando estábamos realizando las entregas de un producto que utilizaba un repositorio Subversion, como medio para identificar la entrega utilizábamos la hora a la que se había subido el último cambio. Pregunté que por qué no se etiquetaba dicha versión y se utilizaba como referencia dicha etiqueta y la respuesta fue que consumía muchos recursos etiquetar y hacer ramas en Subversion. Esto era contrario a lo que yo tenía entendido pero no tenía información con la que pudiese contrastar dicha afirmación. Y como lo mejor que puedes hacer en esta vida es ser curioso, decidí poner a prueba a un servidor Subversion que instalé en casa, y así de paso ayudar a otras personas en su toma de decisiones con respecto al uso de repositorios Subversion, pues éste es el motivo último de estos artículos.

Por supuesto quiero dejar claro una cosa, y es que los resultados obtenidos en este artículo son válidos para la configuración que yo he realizado en mi servidor Subversion, que he intentado sea lo más estándar posible con las versiones de software utilizadas. Cada cual puede realizar las mismas pruebas en su propia instalación y comprobar si se ajustan o no a las mías.

Objetivo

Nuestro objetivo es instalar un servidor Subversion, accesible mediante el protocolo HTTP en nuestro PC local, de manera que lo podamos llevar en un dispositivo de almacenamiento móvil, es decir un servidor "Portable", de manera que podamos ejecutarlo en cualquier PC nuestro. Así, el servidor que vamos a instalar contendrá:

- Los binarios necesarios para ejecutar el servidor Subversion por HTTPD

Histórico de NOTICIAS

Últimos Tutoriales

Liquibase- Incorporación del histórico de cambios en una BBDD existente

Cómo subir tutoriales a Adictos

Spring + REST + JSON = SOAUI

Redescubriendo el Agilismo

CAS: Validador personalizado

Últimos Tutoriales del Autor

Instalación de subversion

Gestión de los Requisitos (y II) : los casos de uso

Gestión de los requisitos

Introducción a JSF Java

Indentación del código fuente

Síguenos a través de:

- El repositorio inicial donde se almacenará la información que queremos guardar bajo un control de versiones.

Requisitos

Un servidor Subversion instalado tal y como se describe en mi artículo anterior dedicado a la instalación y configuración de un servidor Subversion.

Rendimiento de los repositorios Subversion

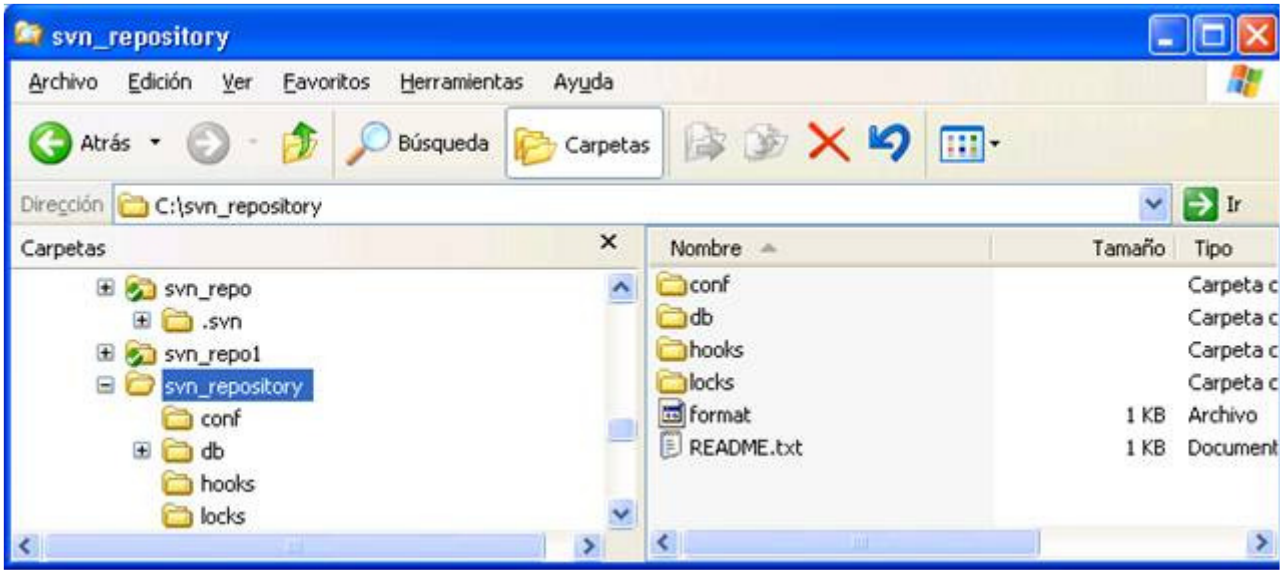
Subversion promete un buen rendimiento en el manejo de los repositorios, con una gran facilidad para manejar ficheros binarios así como operaciones de copia, movimiento, ramas y etiquetas poco costosas.
Vamos a comprobar si esto es cierto, con unas sencillas pruebas.

Pruebas con el control de versiones Subversion

Vamos a someter al repositorio Subversion y al servidor a una pequeña batería de pruebas que nos ayudará a entender cómo gestiona el servidor el espacio en disco ocupado por el repositorio y cómo realiza las transferencias de ficheros entre el cliente y el servidor.

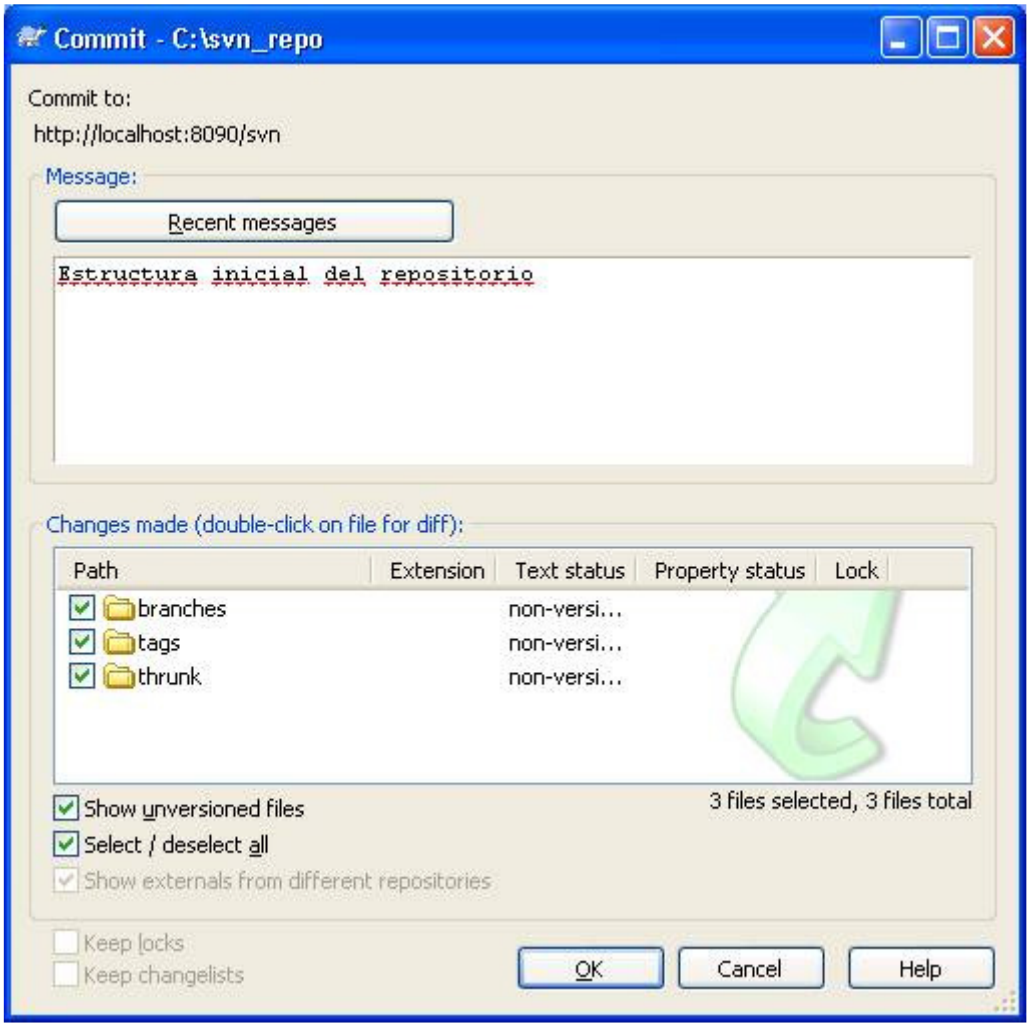
Prueba 1: carga de una carpeta completa de código fuente

Tras haber inicializado el repositorio, la carpeta del repositorio c:\svn_repository contiene una serie de archivos y carpetas que definen la estructura y los datos iniciales del repositorio.

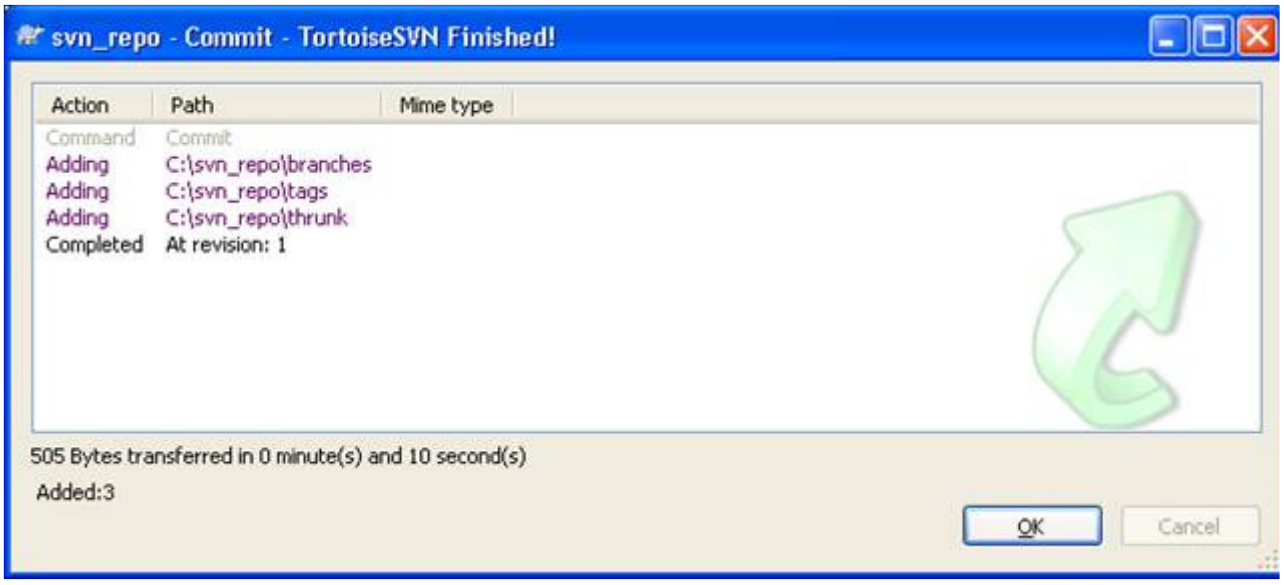


El tamaño inicial de esta carpeta es de unos 104 kbytes (dependerá un poco de cada instalación, nos interesan los números aproximados para nuestras pruebas)
Ahora creo las tres carpetas thrunk, branches y tags en c:\svn_repo y uso la opción del TortoiseSVN "SVN commit", pulsando con el botón secundario del ratón sobre la carpeta c:\svn_repo

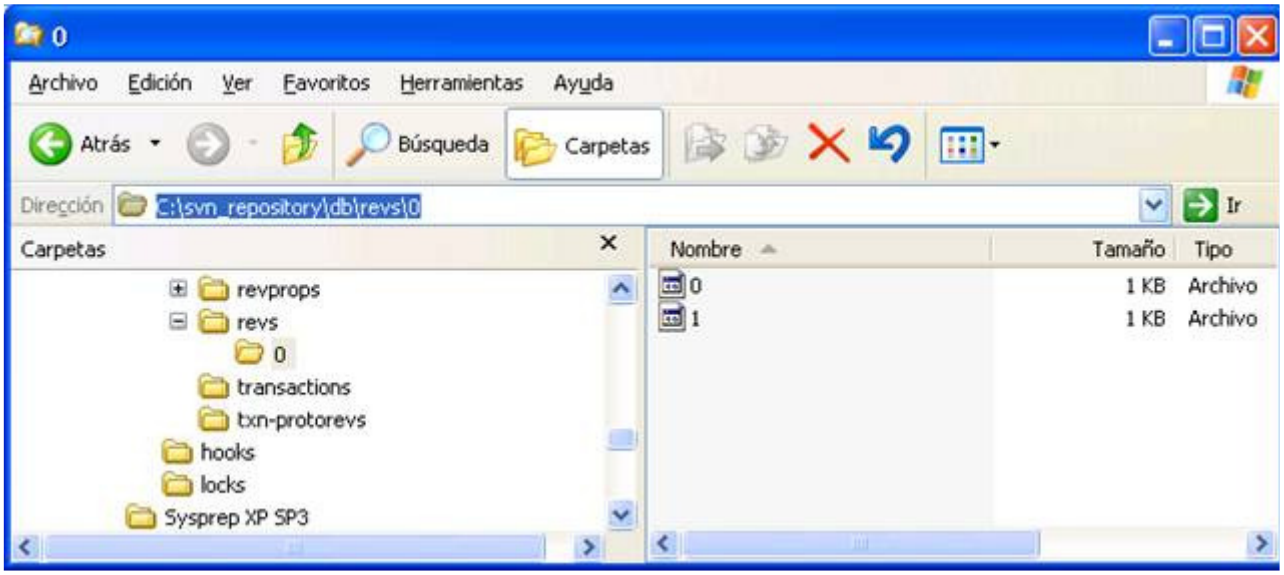
Nota: en adelante no indicaré que los comandos de Tortoise SVN se obtienen con el botón secundario del ratón sobre la carpeta o fichero de interés…



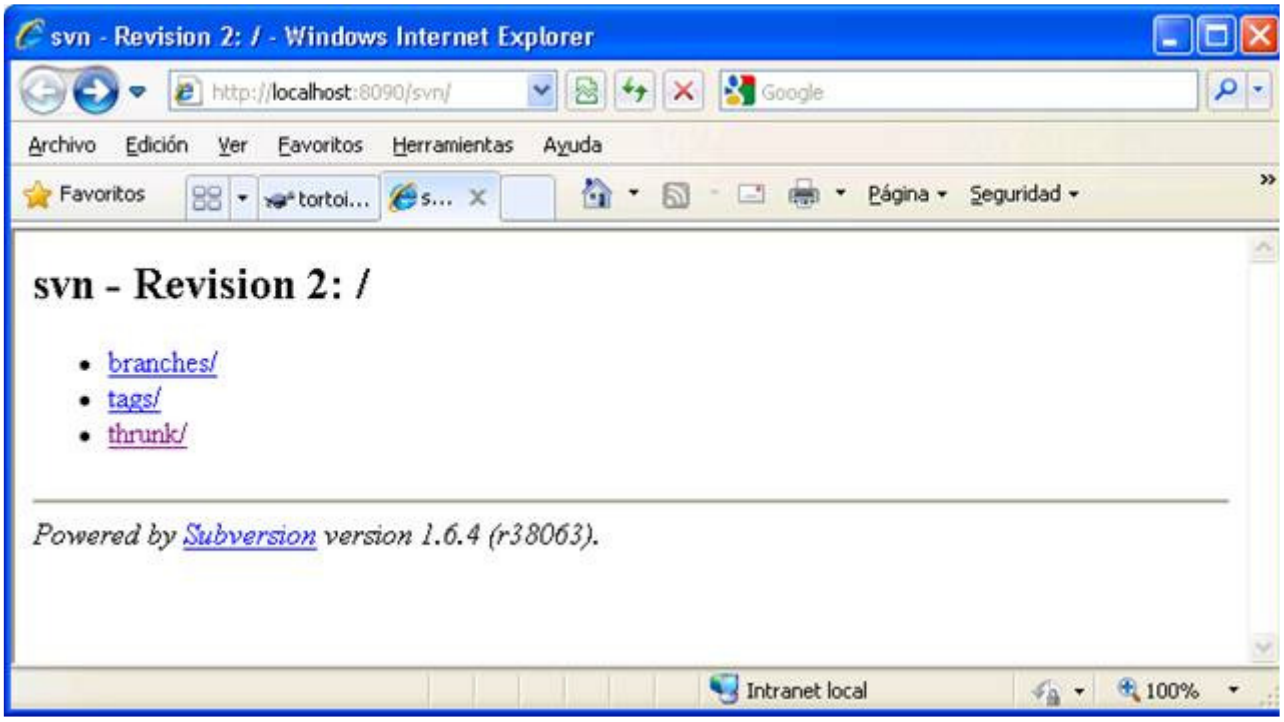
Marco las tres carpetas, pues no las he añadido previamente con "Tortoise SVN…\Add…"Pulso OK



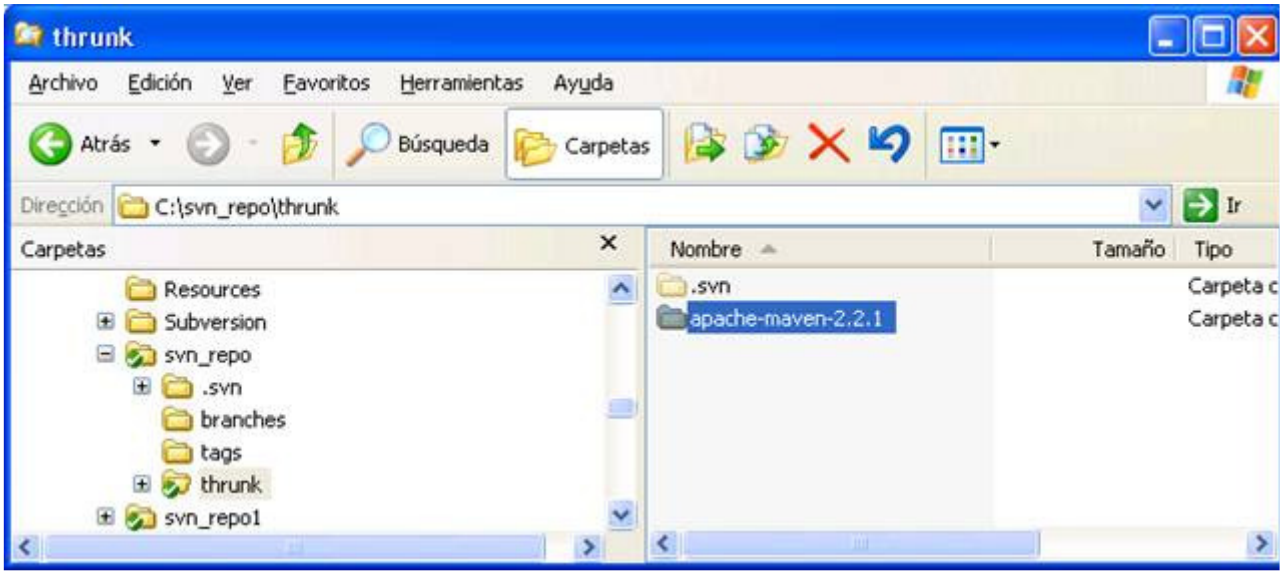
Ahora mi carpeta c:\svn_repository contiene en la carpeta C:\svn_repository\db\revs\0 una nueva revisión (la 1)



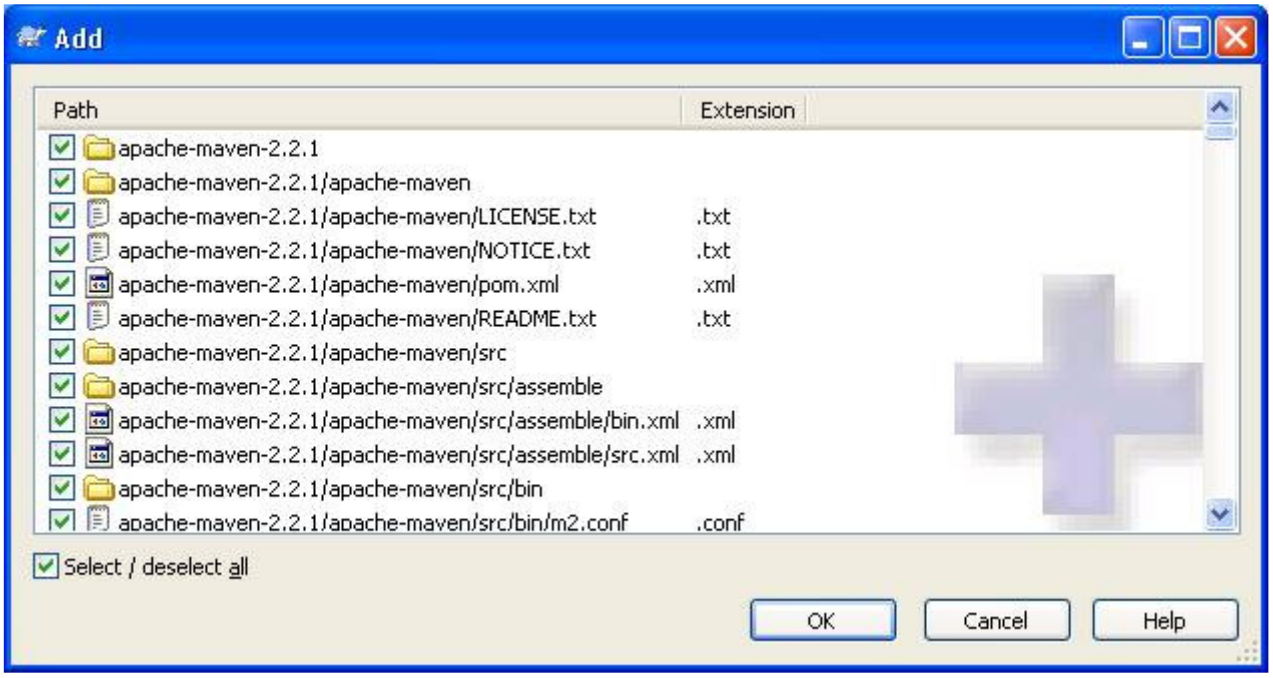
Si accedemos al repositorio con el navegador veremos reflejados los cambios



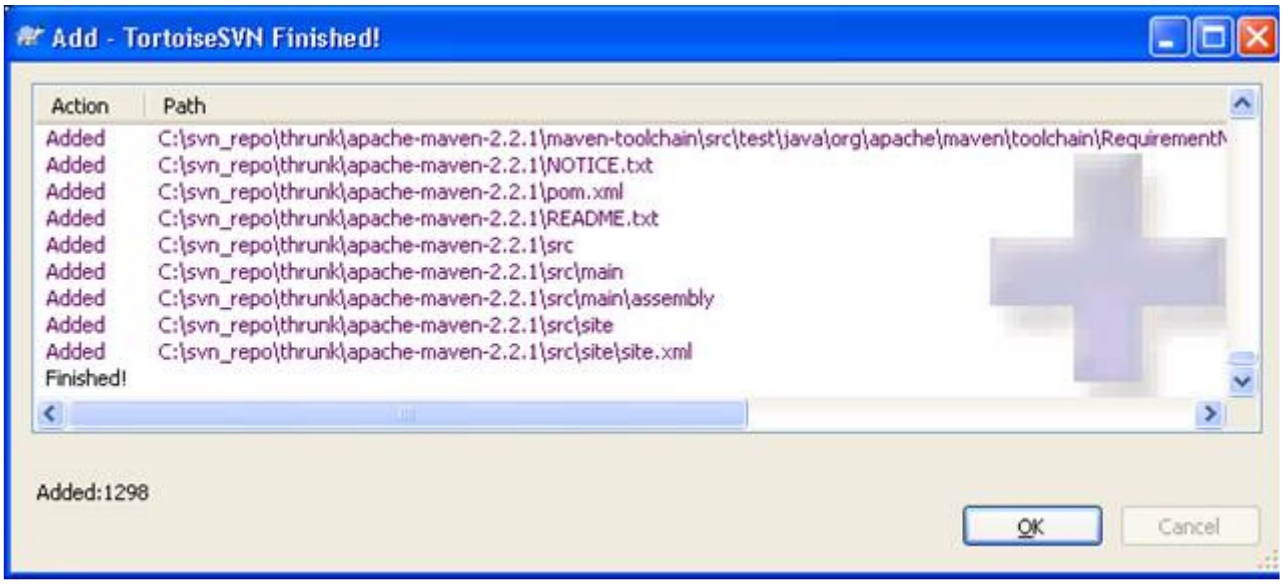
Ahora procedo a añadir una carpeta de código fuente. Para ello voy a usar el fichero apache-maven-2.2.1-src.zip que contiene el código fuente del Maven 2.2 (lo tenéis en www.apache.org): Descomprimo este fichero en una carpeta y copio el contenido completo a c:\svn_repo\thrunk (creo la carpeta si es necesario)



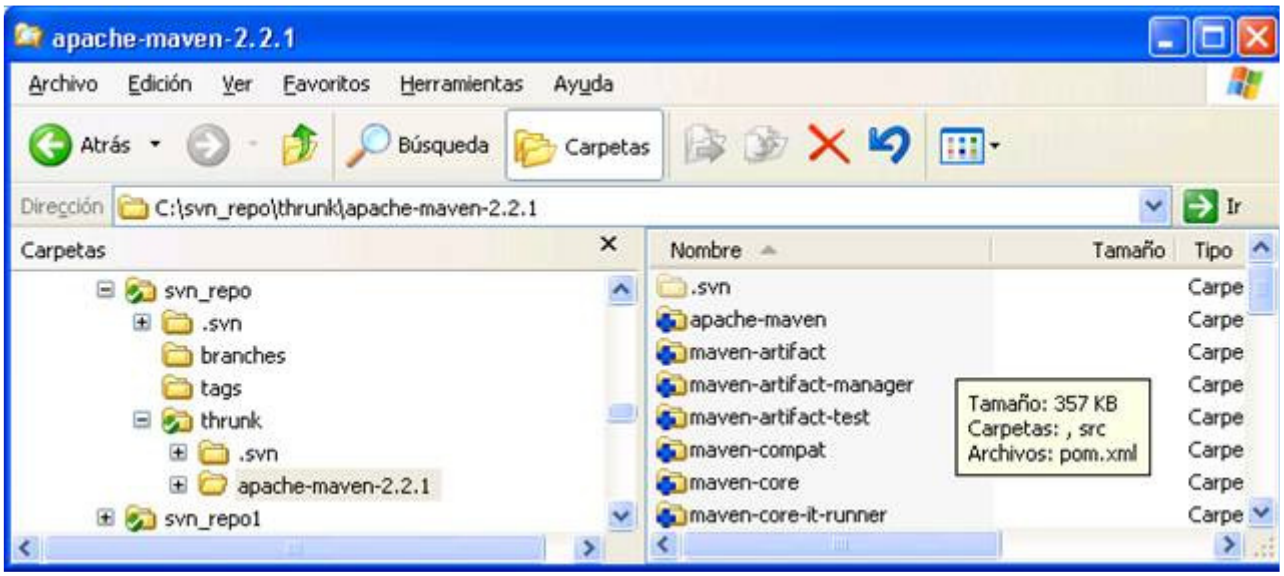
Ahora procedemos a añadir la nueva carpeta de Maven al control de versiones. Para ello lo primero es usar la opción "Tortoise SVN… \ Add …"



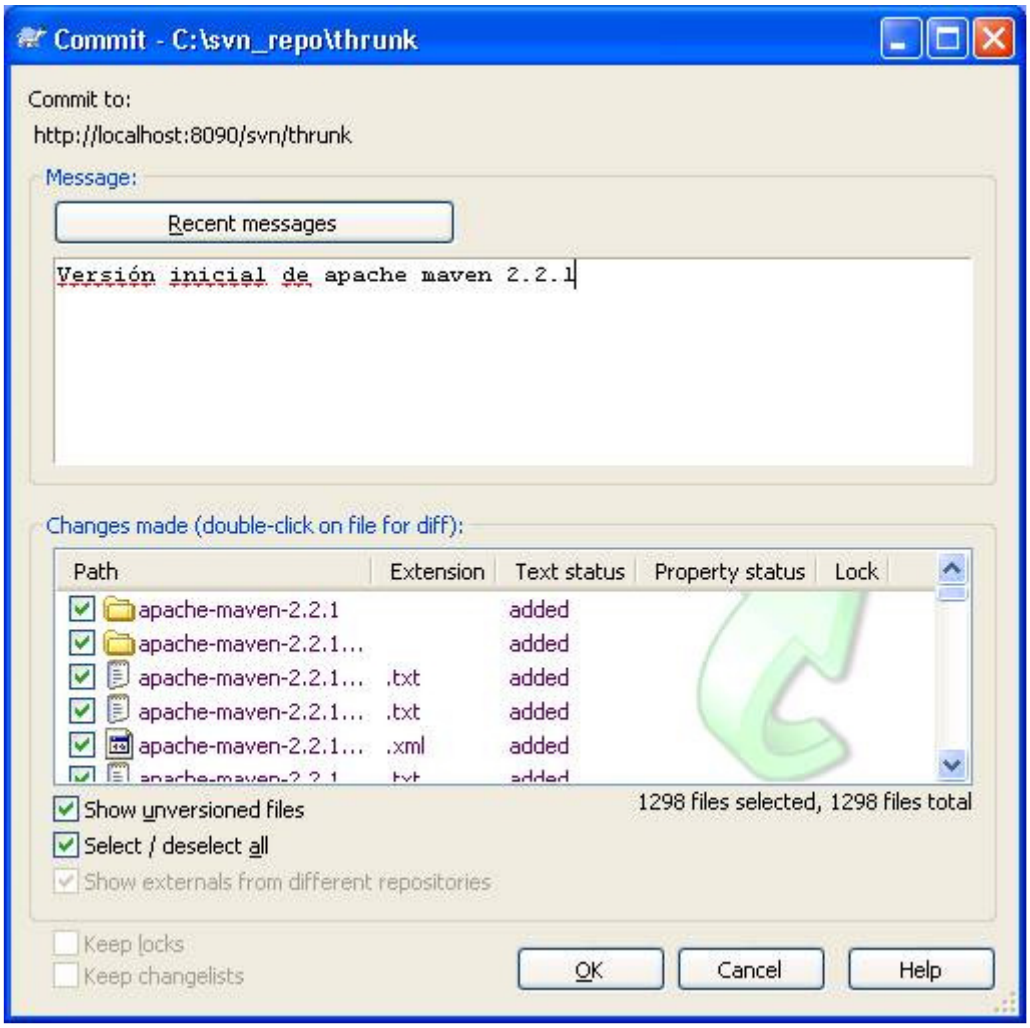
Subversion no utiliza la distinción entre binarios y texto que usa CVS, sino que cada fichero tiene asociado un tipo MIME, que es el usado por el servidor Web cuando el repositorio se consulta a través de un navegador. Si alguna extensión no la conoce, le añade application/octet-stream y a correr…
Pulsamos OK. Tras un rato de trabajo nos avisará de que se han añadido 1298 elementos (ficheros y carpetas), que ocupan unos 3,8Mbytes, frente a los 3,5Mb que ocupa el fichero zip descomprimido, ya que se añaden carpetas ocultas .svn.



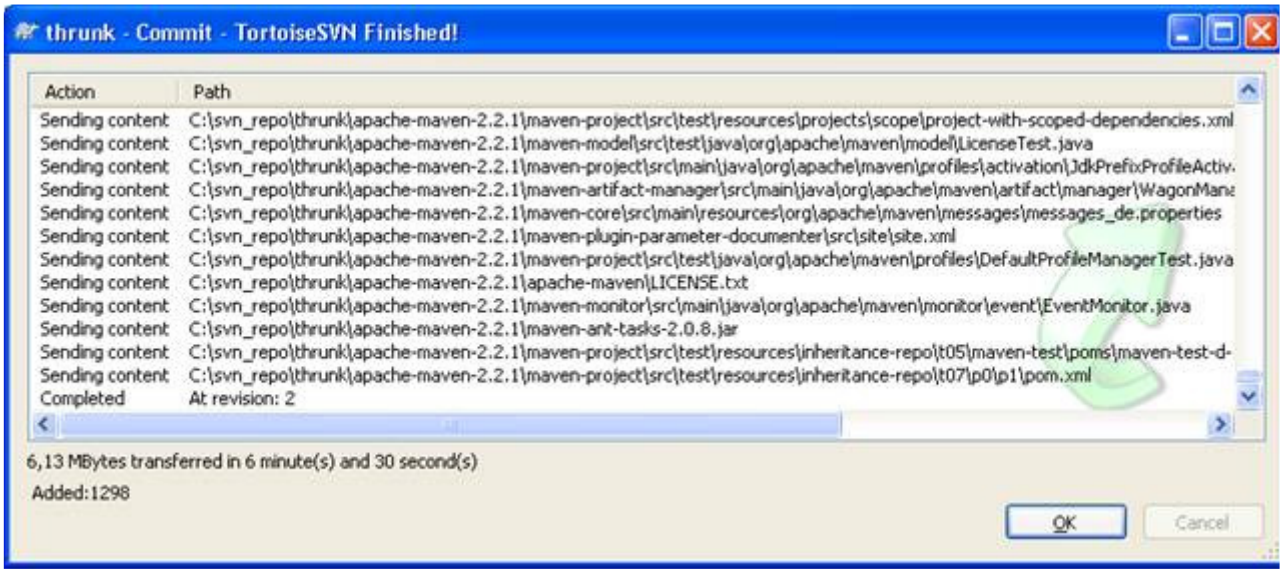
Ahora cada fichero o carpeta añadido está marcado con una cruz azul



Ahora procedemos a subir los cambios al control de versiones, con SVN commit sobre la carpeta a thrunk

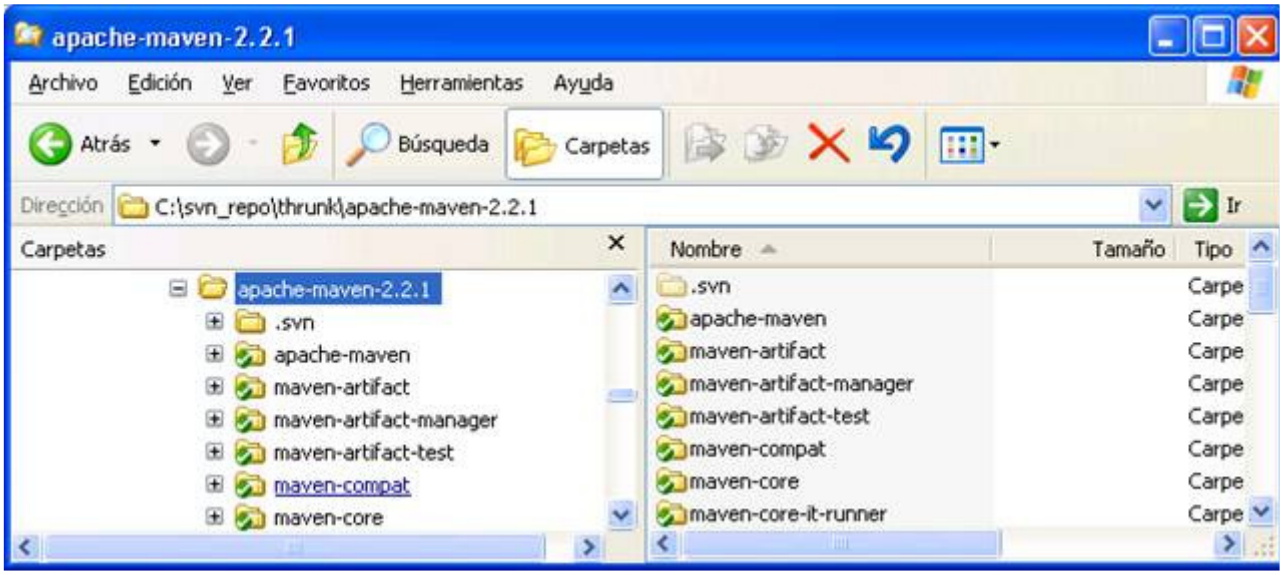


Pulsamos OK

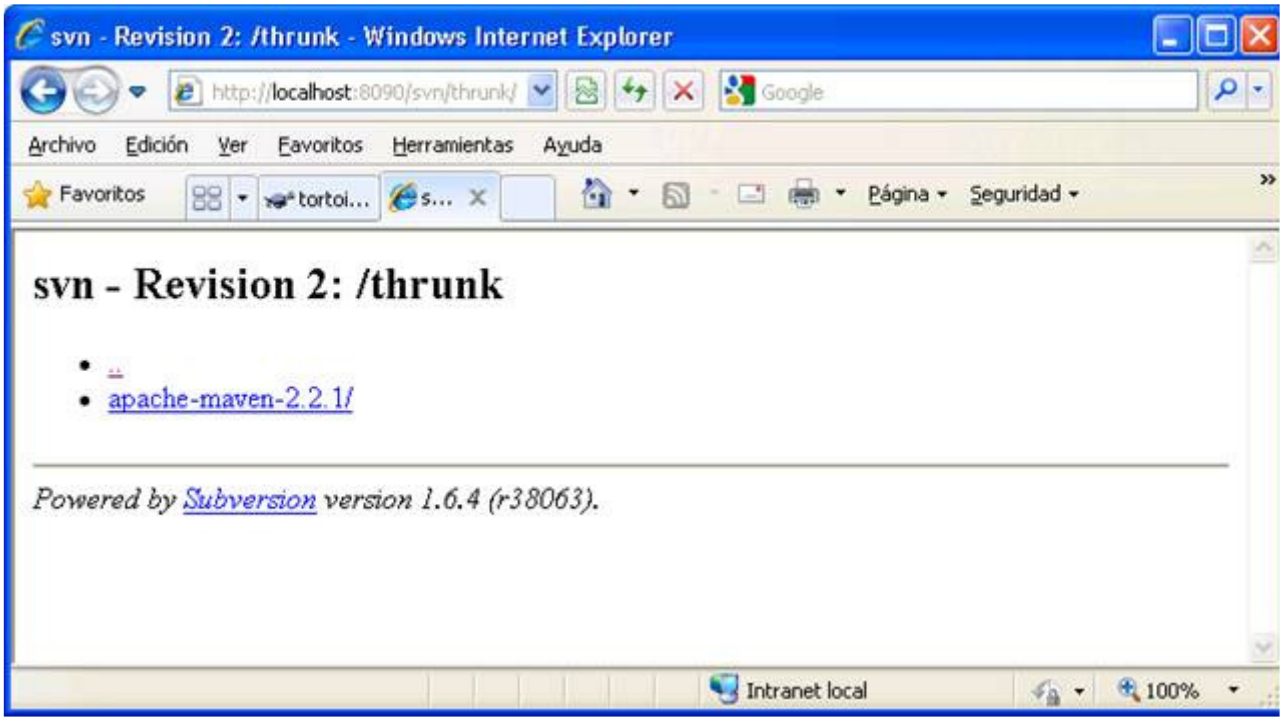


El cliente Tortoise SVN nos informa de que se han transferido 6Mbytes con el servidor. Esto se debe a que las primeras subidas son pesadas. Además se han transferido 1298 objetos, como ya habíamos obtenido con el comando SVN add.

La carpeta en nuestra copia local del repositorio ahora indica con unos tics verdes que todo está correcto y actualizado a la última revisión

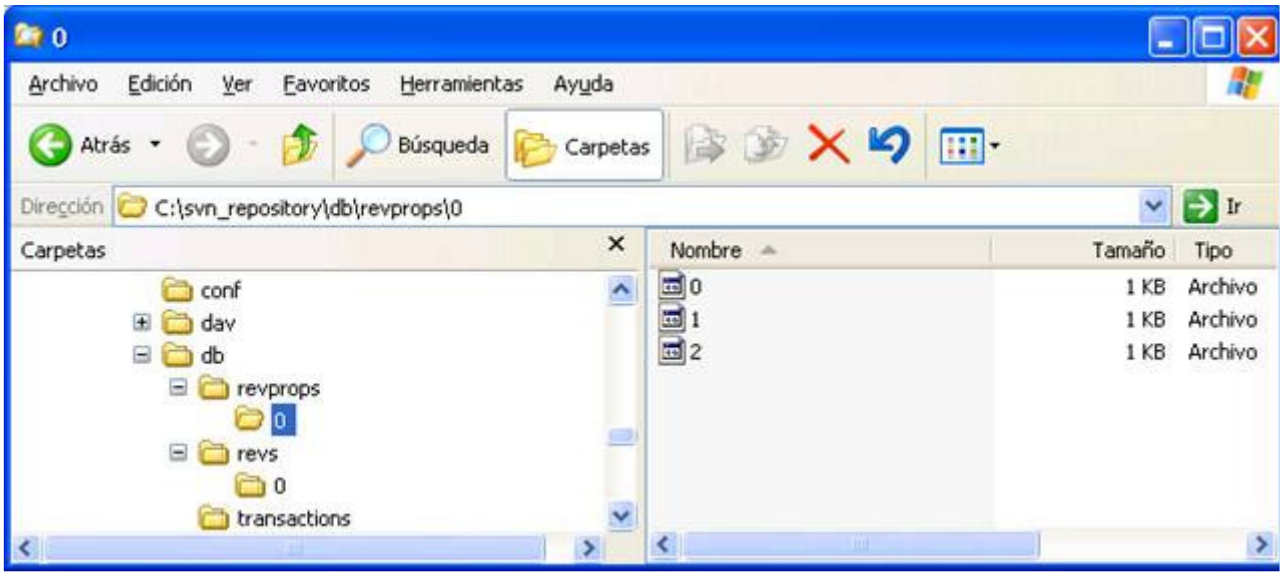


Ahora el control de versiones está en la revisión 2, como podemos ver si accedemos al repositorio con el navegador



Sin embargo, ahora la carpeta C:\svn_repo\thrunk\apache-maven-2.2.1 ocupa casi 8Mbytes. Esto se debe a que Tortoise SVN genera copias completas de los ficheros añadidos al control de versiones en las carpetas .svn.

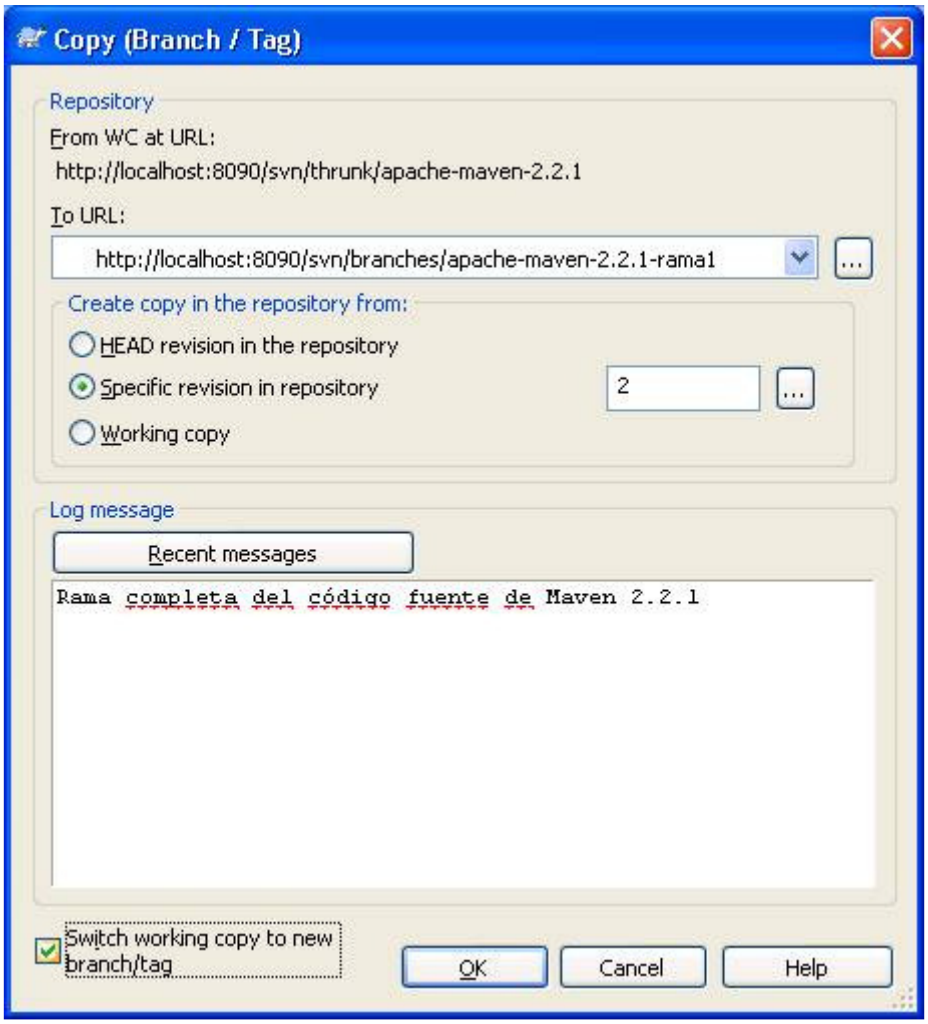
Por su parte en el repositorio c:\svn_repository simplemente se ha creado un nuevo archivo, con los datos de la revisión 2



Con esto vemos que el servidor Subversion almacena eficientemente todos los datos subidos en una revisión, pues coincide casi con lo que ocupa el zip de apache-maven-2.2.1

Prueba 2. Creación de una rama del desarrollo

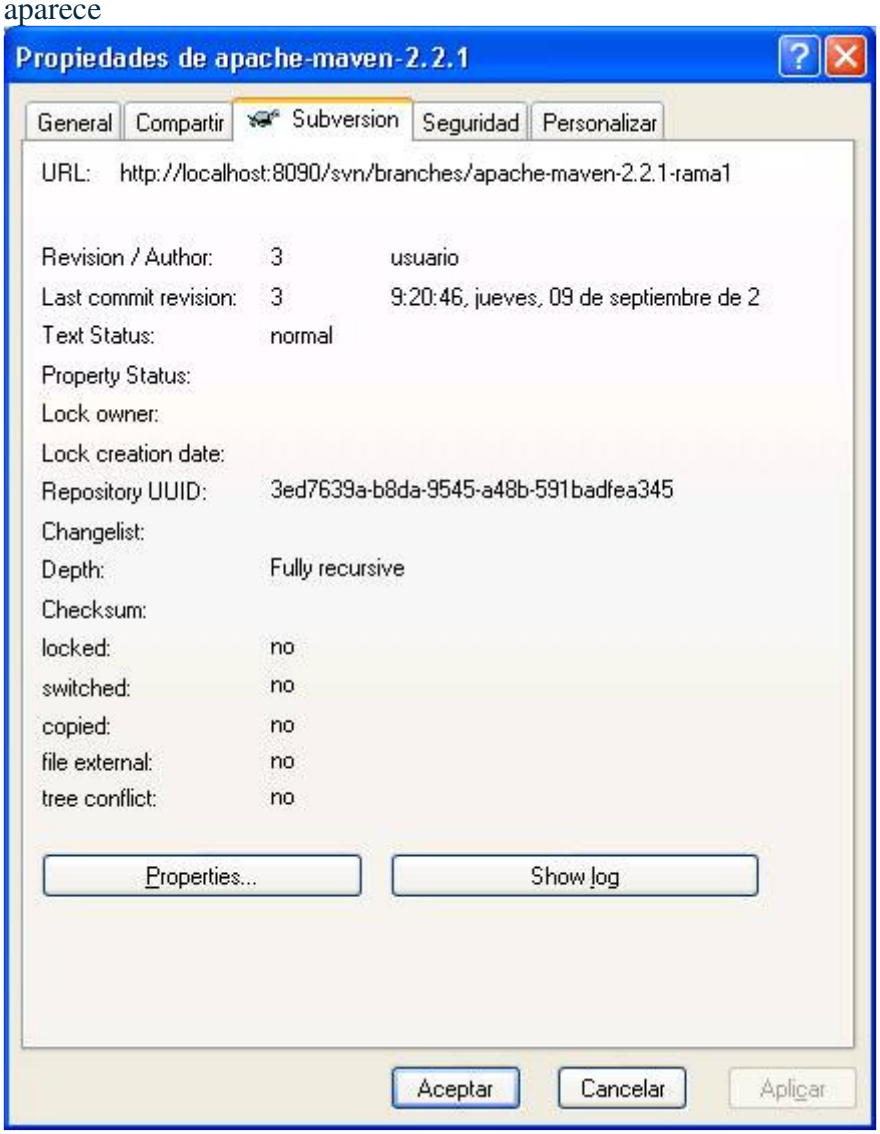
Ahora vamos a ver cómo se realiza una rama del desarrollo con Subversion y el impacto que tiene en el repositorio.
Seleccionamos la carpeta apache-maven-2.2.1 y pulsamos en “Tortoise SVN \ branch/tag…”



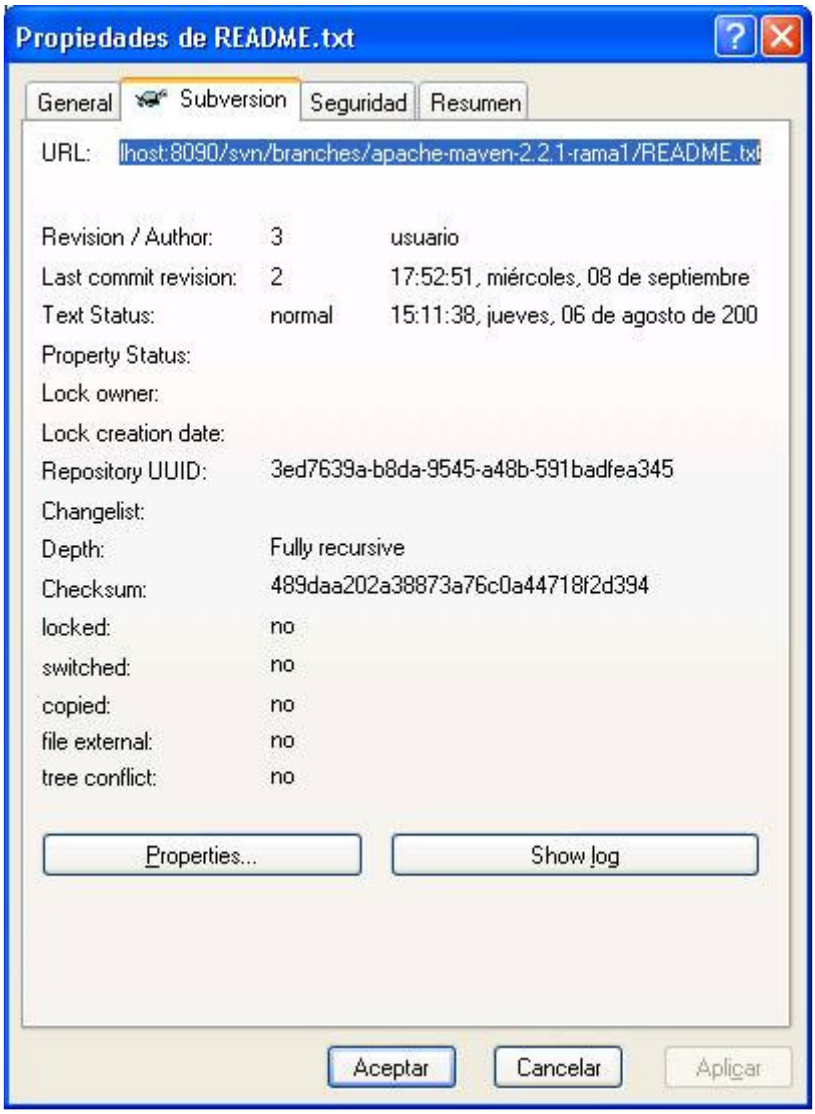
Voy a crear una rama en http://localhost:8090/svn/branches/apache-maven-2.2.1-rama1 . Observamos que no hay diferencia en subversion entre rama y etiqueta. También he marcado la casilla para cambiar automáticamente a la nueva rama en la copia local. Pulsamos OK

Apenas ha transferido información. Y además ha sido rapidísimo.
Veamos qué ha ocurrido en el repositorio local

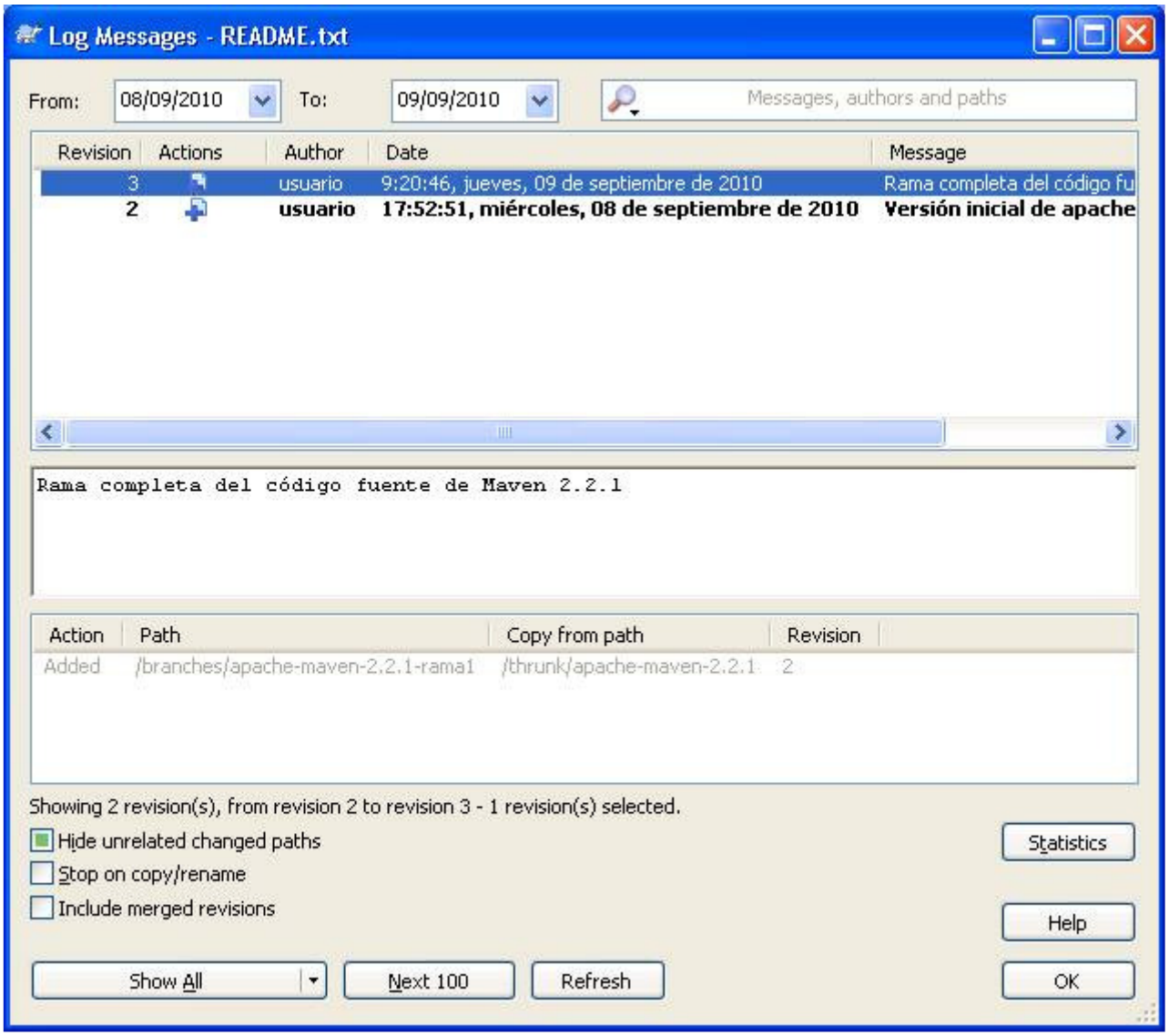
Si pulsamos en las propiedades de thrunk/apache-maven-2-1-1 nos aparece



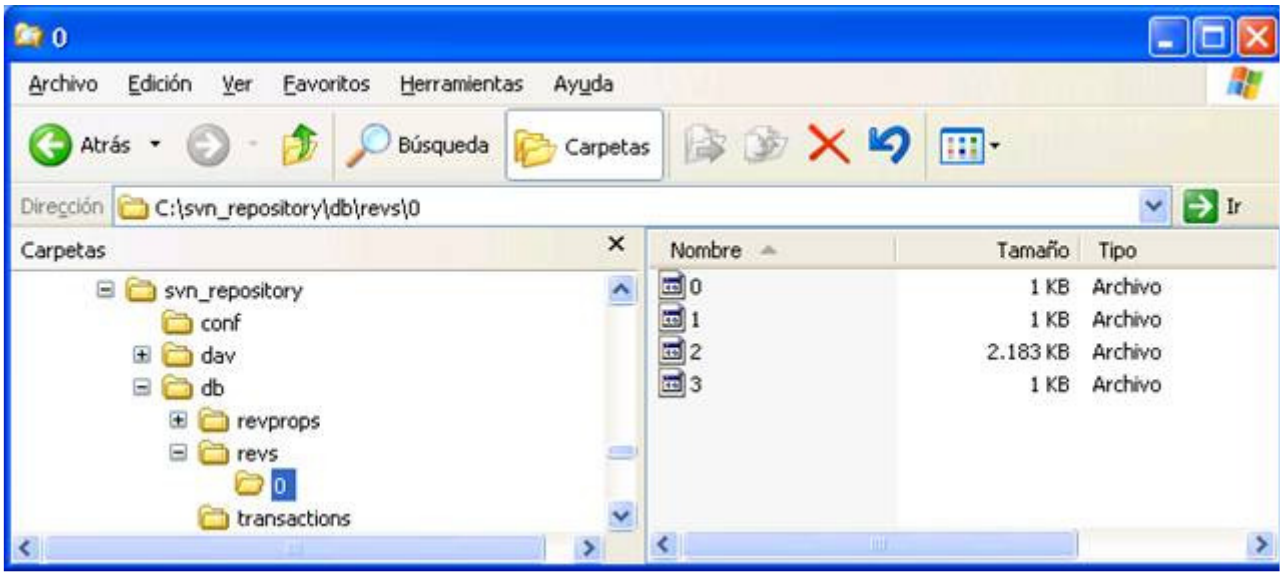
Esto nos indica que esta carpeta ya cambiado de rama. Veamos un archivo de esta rama. Selecciono por ejemplo el fichero README.TXT



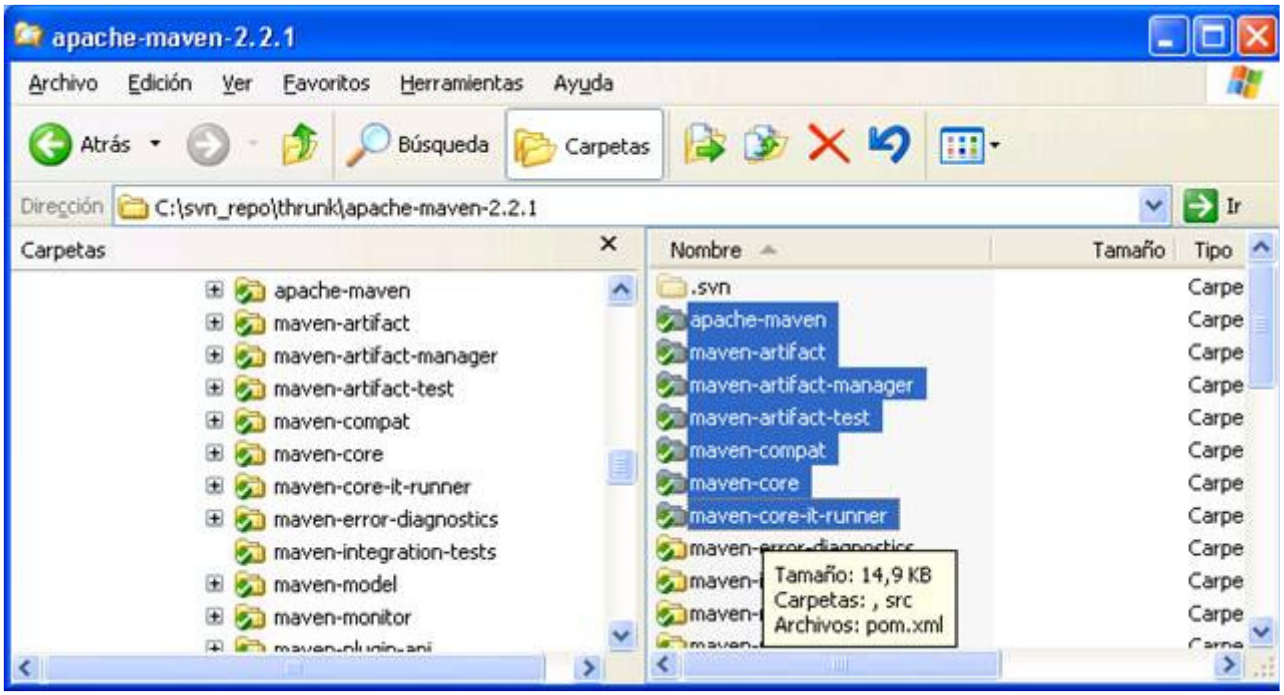
Tal y como cabía esperar también ha cambiado de rama, como lo habríamos hecho si hubiésemos hecho un checkout desde la URL de la rama http://localhost:8090/svn/branches/apache-maven-2.2.1-rama1
Mostramos ahora su histórico de cambios con "Tortoise SVN \ Show log…"



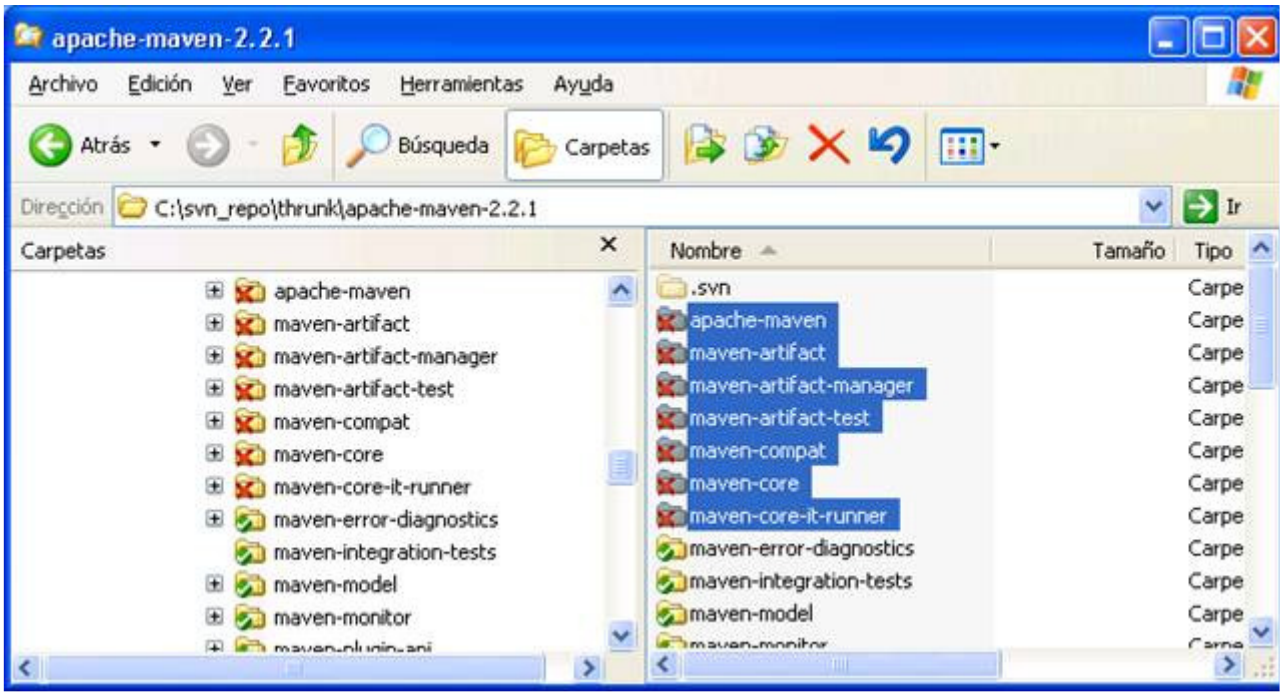
Y ahora lo más interesante, veamos qué ocurre en el repositorio subversion.
En primer lugar vemos que el tamaño total del repositorio apenas ha variado (2,23 Mbytes). Esto se debe a que cuando Subversion crea una rama o etiqueta no copia los ficheros, sino simplemente crea un conjunto de referencias internas a los ficheros antiguos en sus nuevas ubicaciones. Además etiquetas y ramas son tratadas como carpetas normales dentro del repositorio.



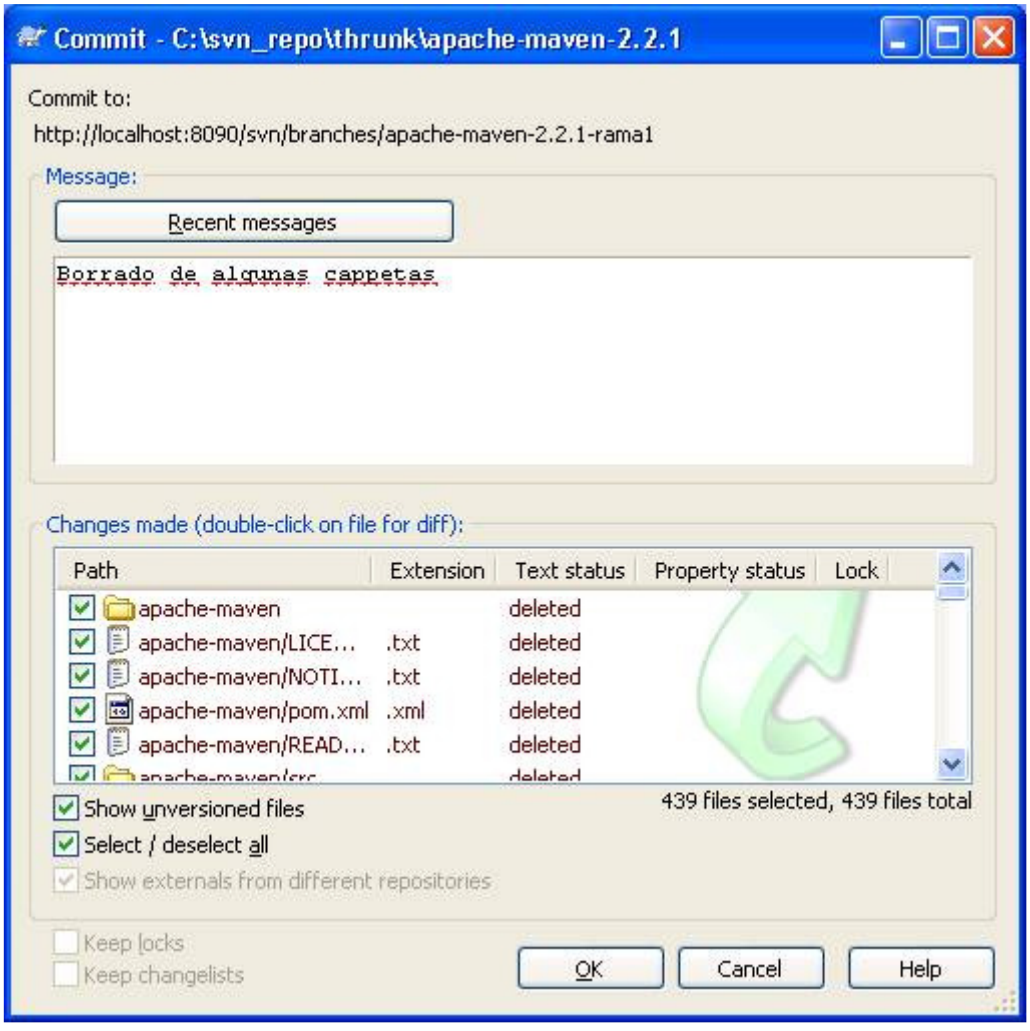
En el sistema de ficheros del repositorio subversion, la operación de creación de la rama apenas ha consumido espacio.
Voy a ver que ocurre si borro algunas carpetas de la rama nueva.
Selecciono primero algunas carpetas:



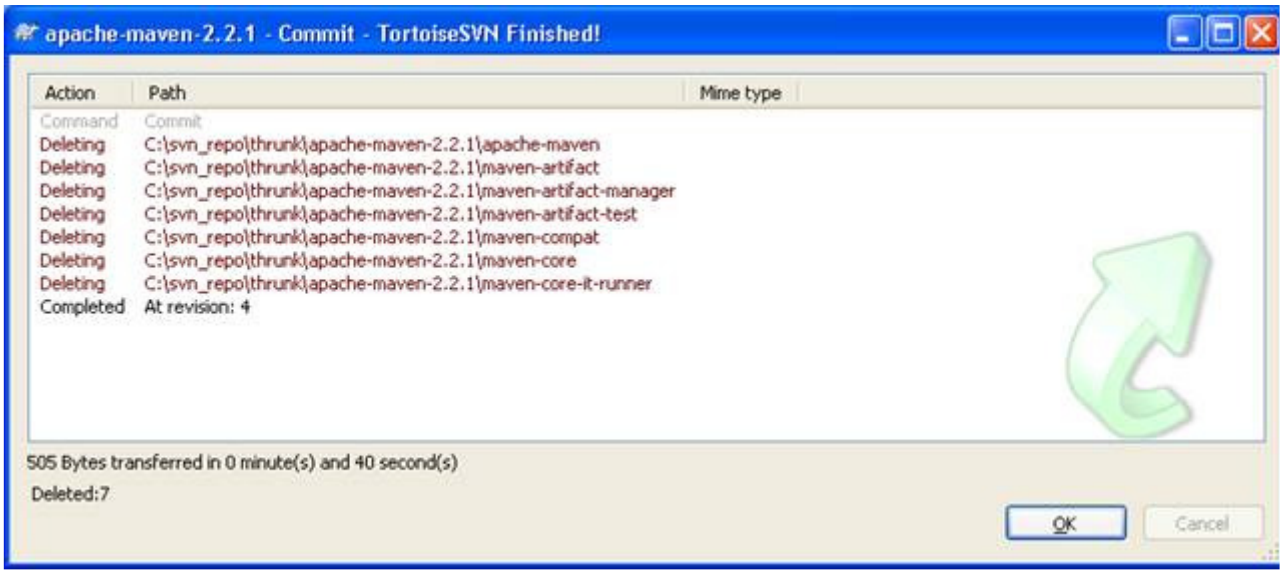
Y utilizo la opción “Tortoise SVN…. \ delete…”



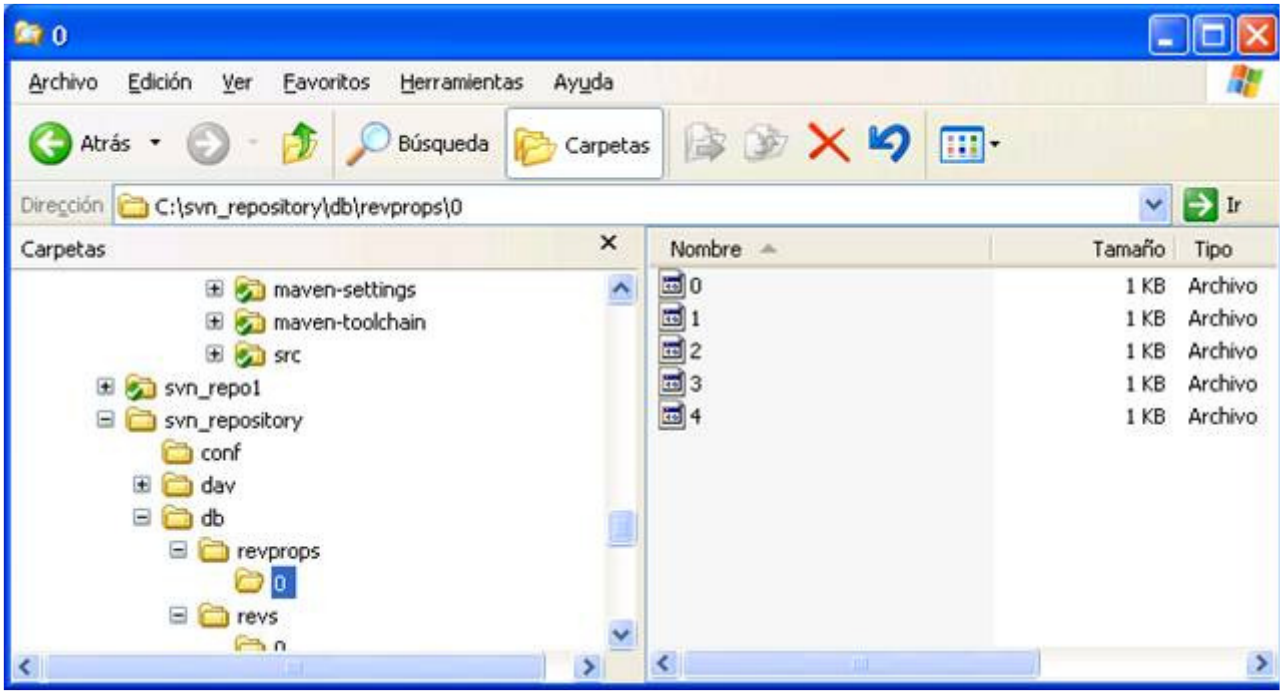
Ahora subo los cambios con “SVN commit” sobre la carpeta apache-maven-2.2.1



Pulso OK

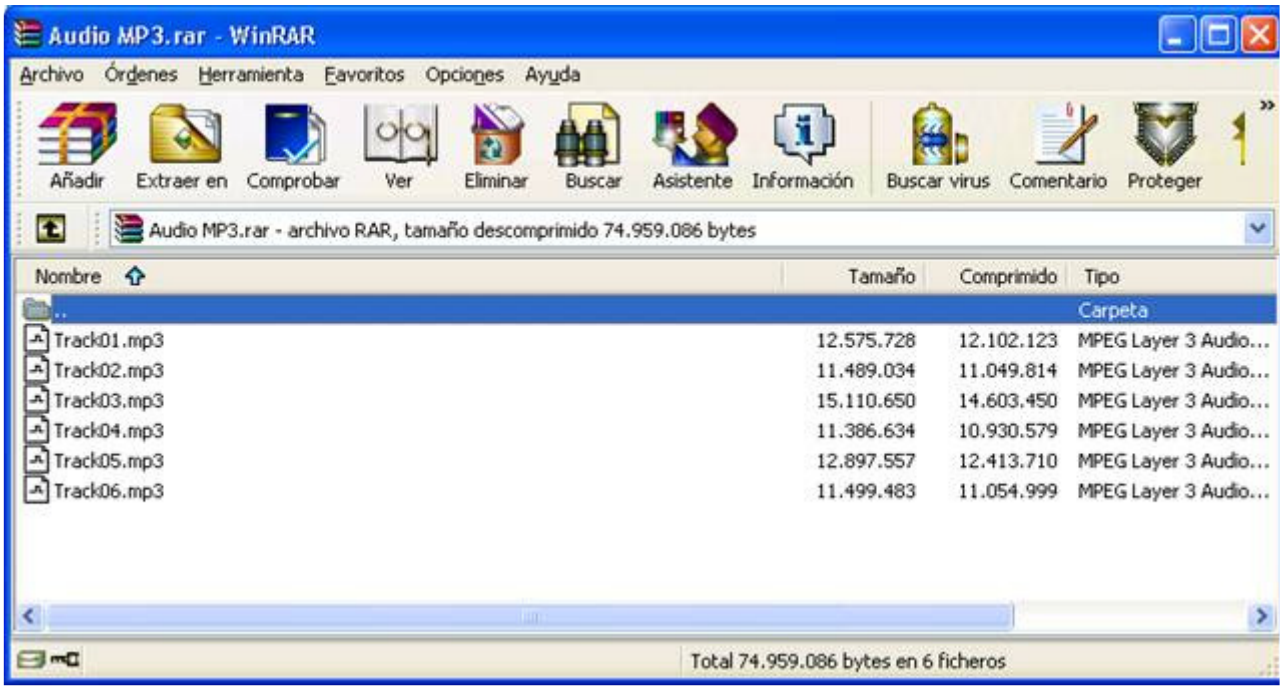


Apneas ha tardado en realizar la operación. Y en cuanto al repositorio ya os podéis imaginar

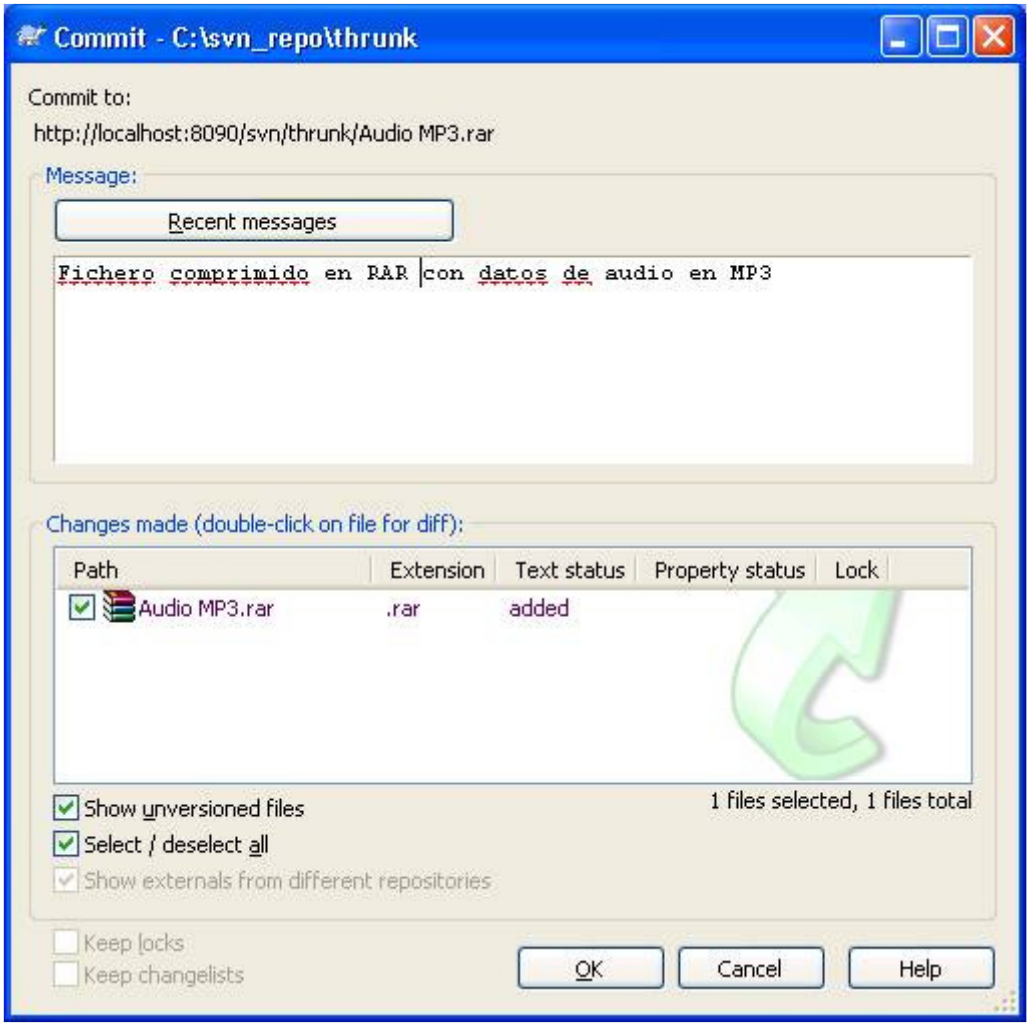


Prueba 3: carga de un conjunto de ficheros binarios de audio MP3 de gran peso

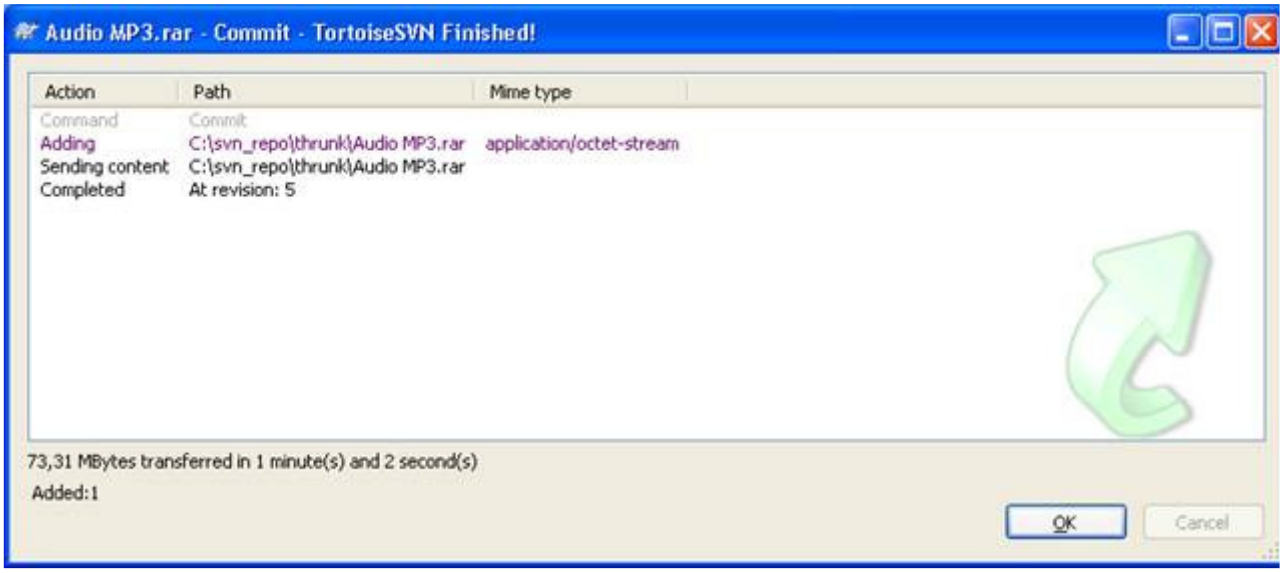
Ahora vamos a realizar una prueba bastante dura para Subversion, que consiste en añadir un fichero comprimido en formato RAR que además contiene un conjunto de pistas de audio.



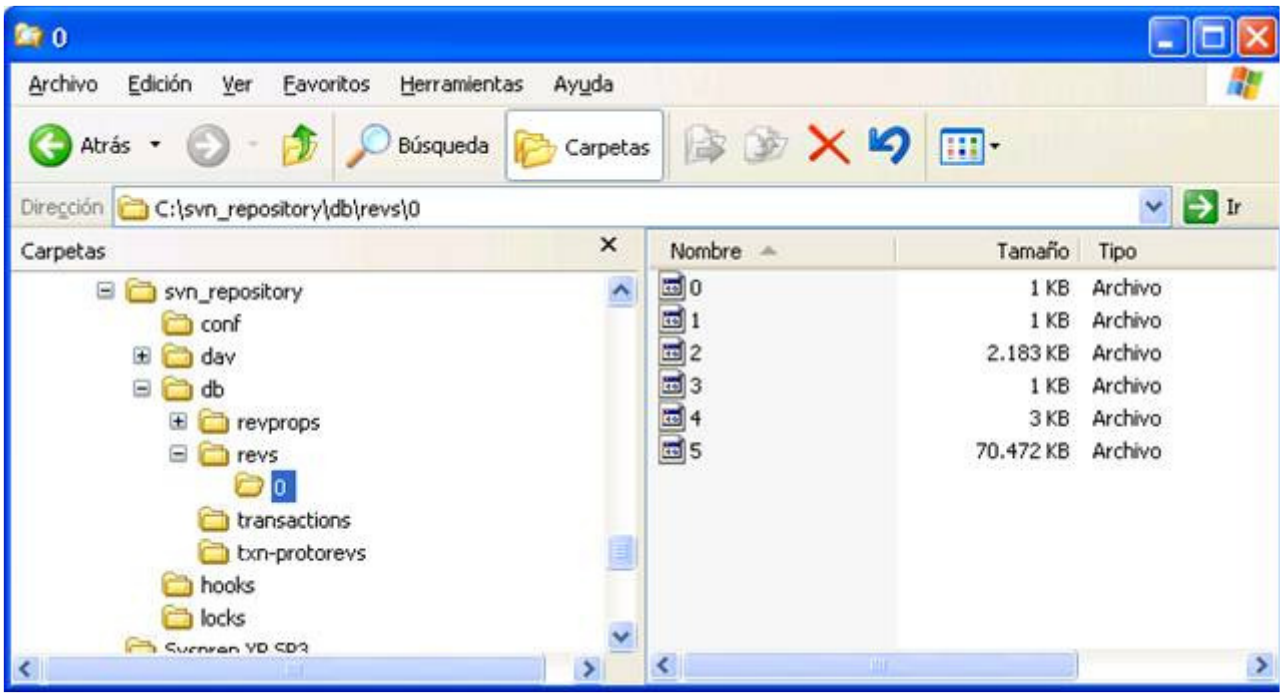
Añadimos este fichero al repositorio en la carpeta thrunk, por ejemplo. Primero usamos "Tortoise SVN \ Add…" y luego "SVN commit…"



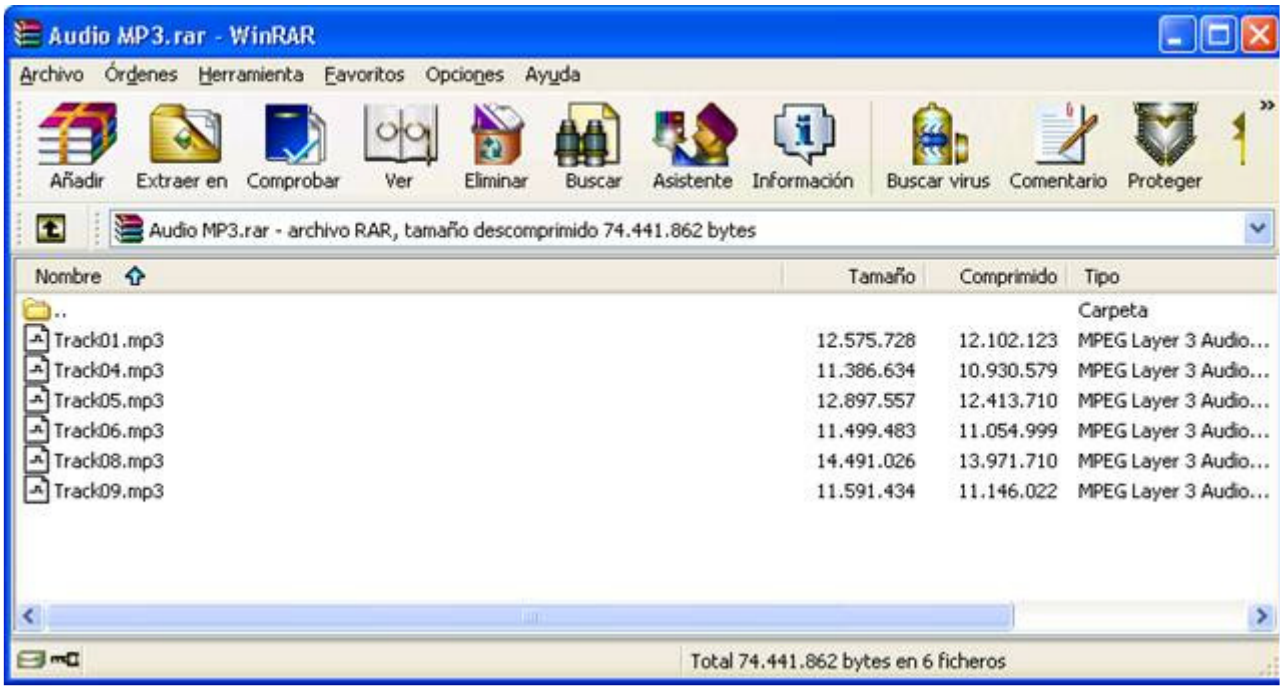
Tras pulsar OK comenzará a enviar el archivo. Tras subir los datos (unos 70Mbytes) se nos muestra el mensaje de confirmación.



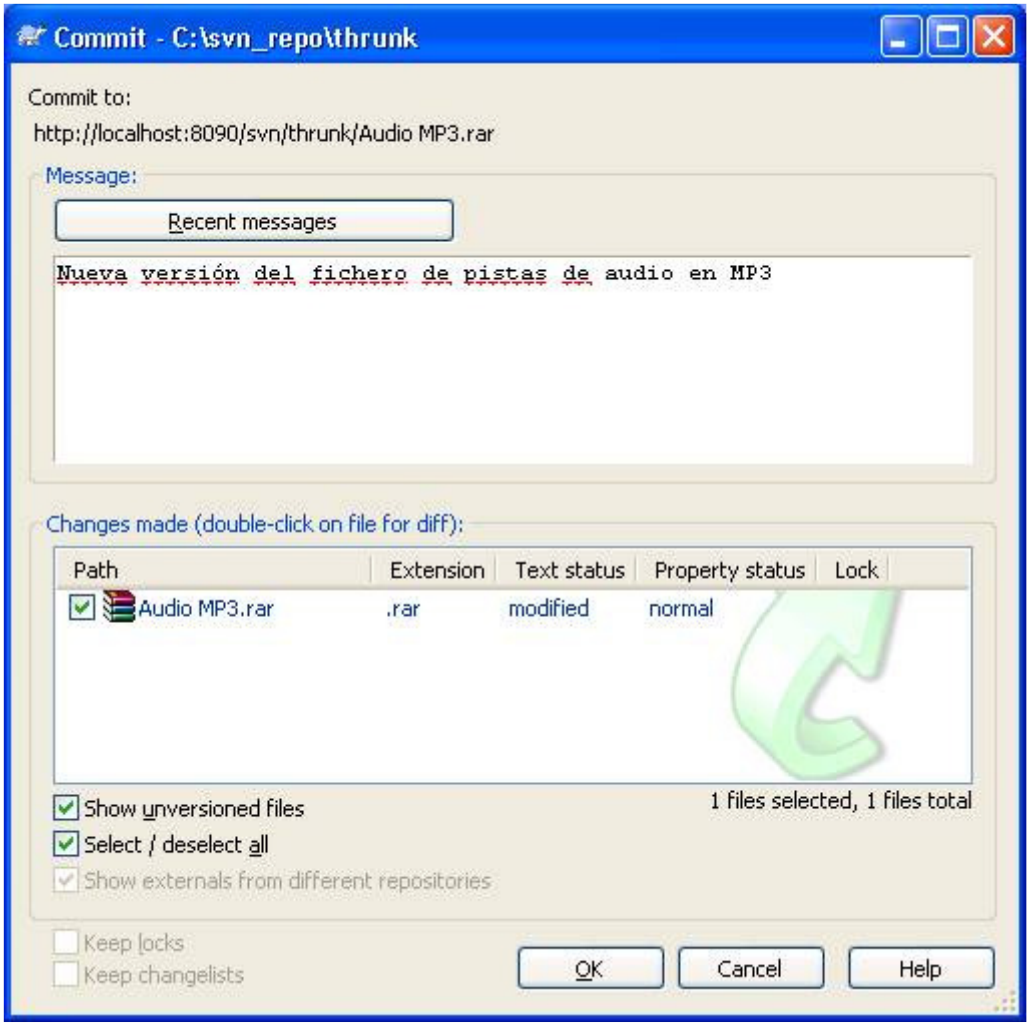
El repositorio ahora contiene ya los datos del fichero enviado



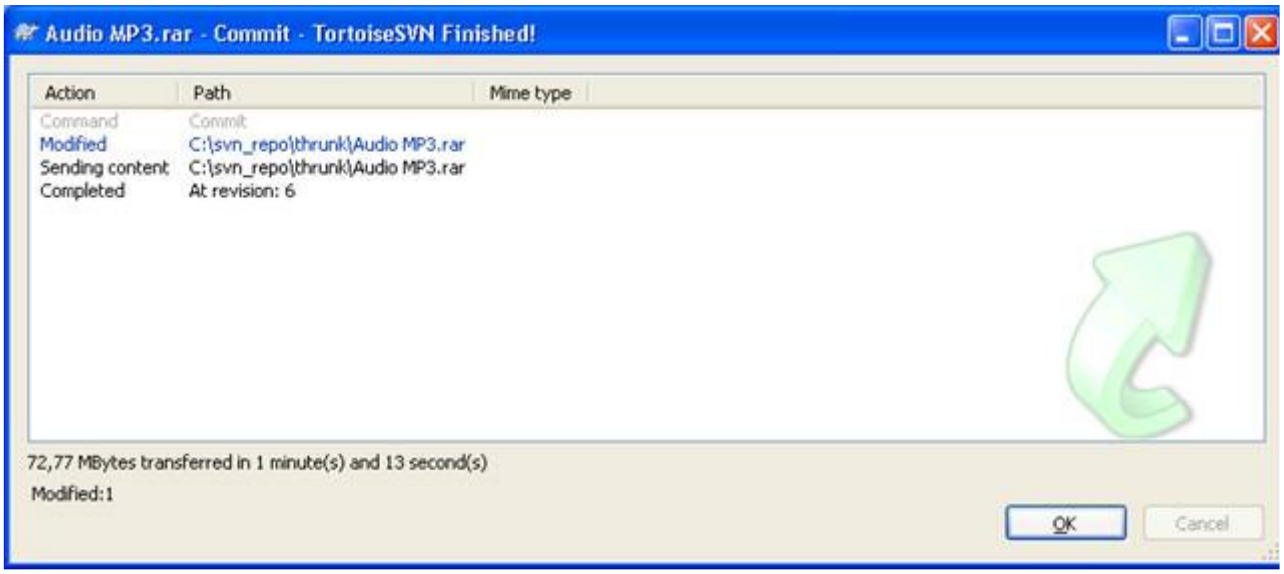
Ahora vamos a realizar el "más difícil todavía". Voy a eliminar un par de pistas del fichero rar y voy a añadir dos pistas nuevas.



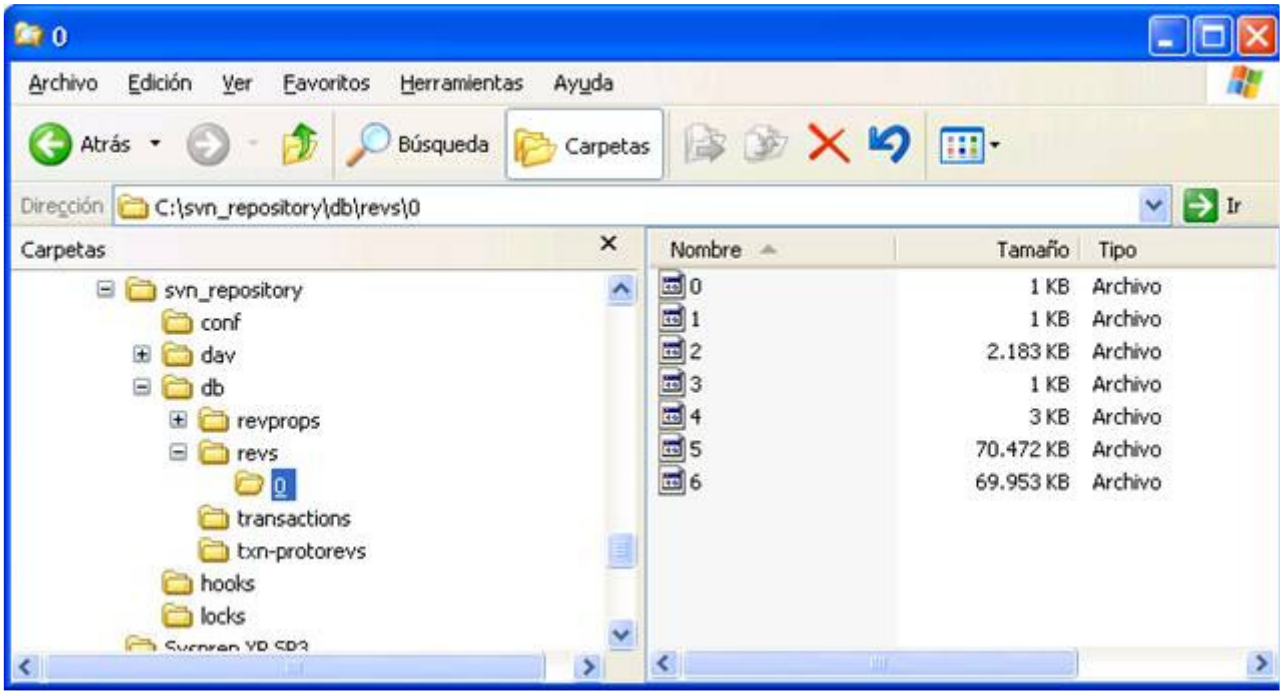
Esta operación es realmente extrema, pues estamos reemplazando par ficheros dentro de un fichero comprimido. Veamos qué ocurre. Hemos eliminado un par de ficheros y hemos añadido dos ficheros de unos 25 Mbytes en total.



Pulsamos OK

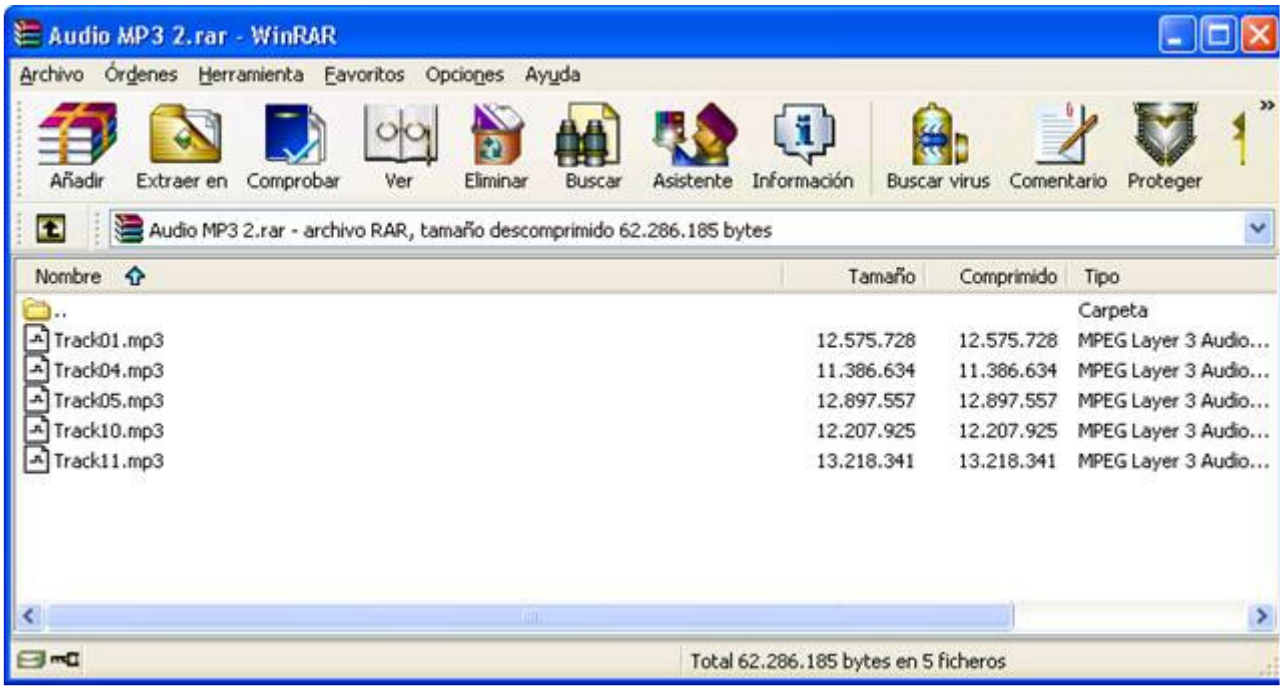


Y veamos el repositorio

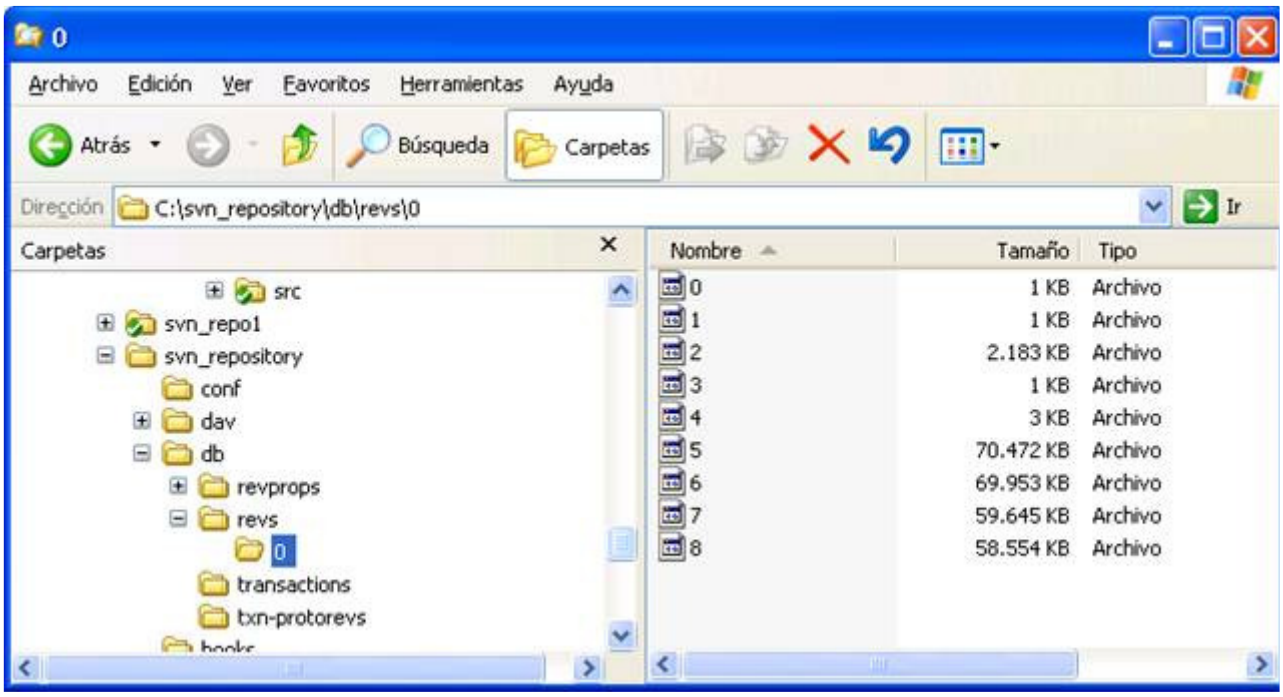


Ha subido de nuevo el archivo completo.

Ahora vamos a hacer otra prueba. Esta vez voy a crear un archivo RAR sin comprimir sólido con 5 ficheros de audio en formato MP3. Subimos el archivo y preparamos una nueva versión

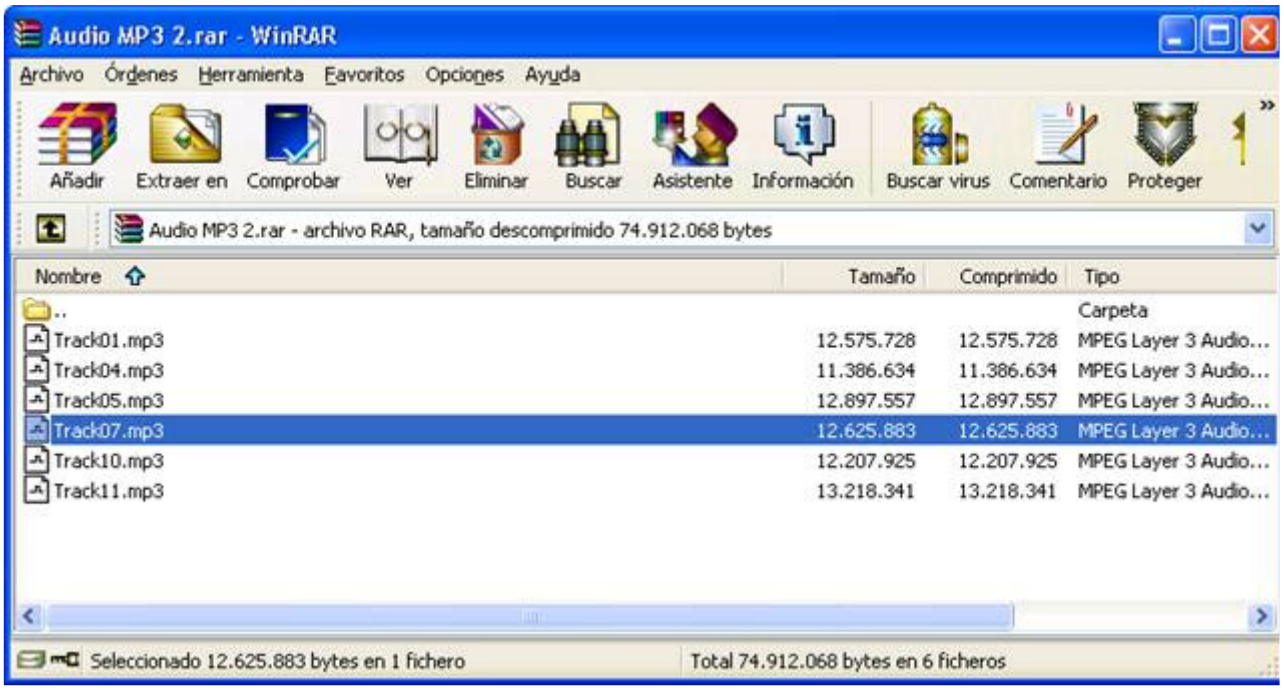


Hemos eliminado las pistas 2 y 3 y añadido la 10 y la 11, sin comprimir.

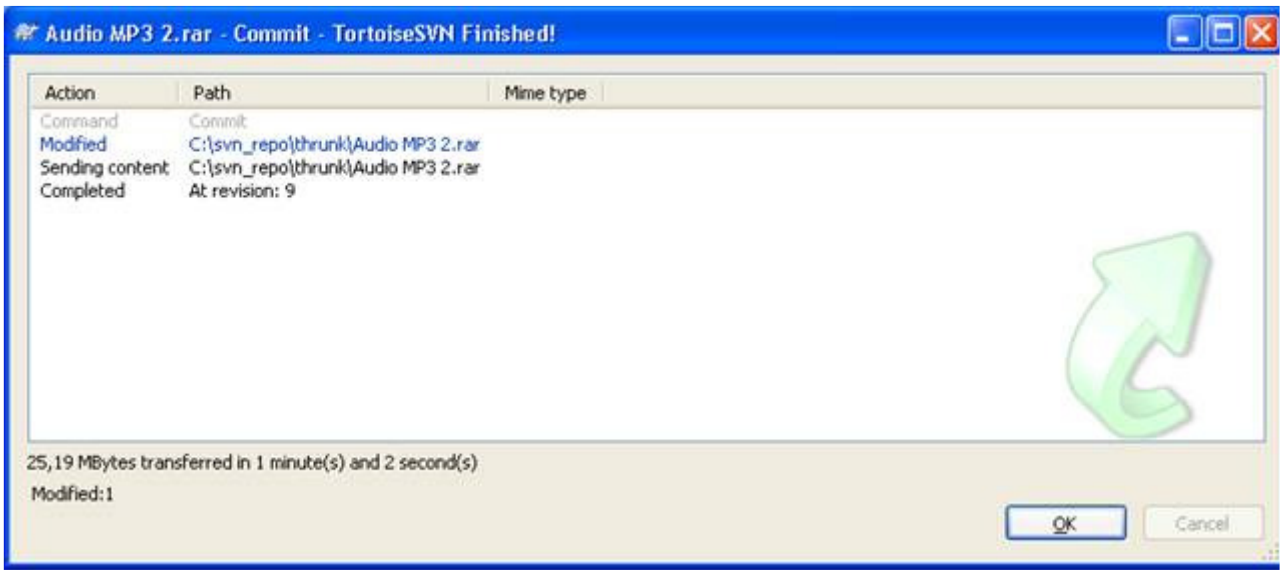


Realmente parece que se suben versiones completas del fichero

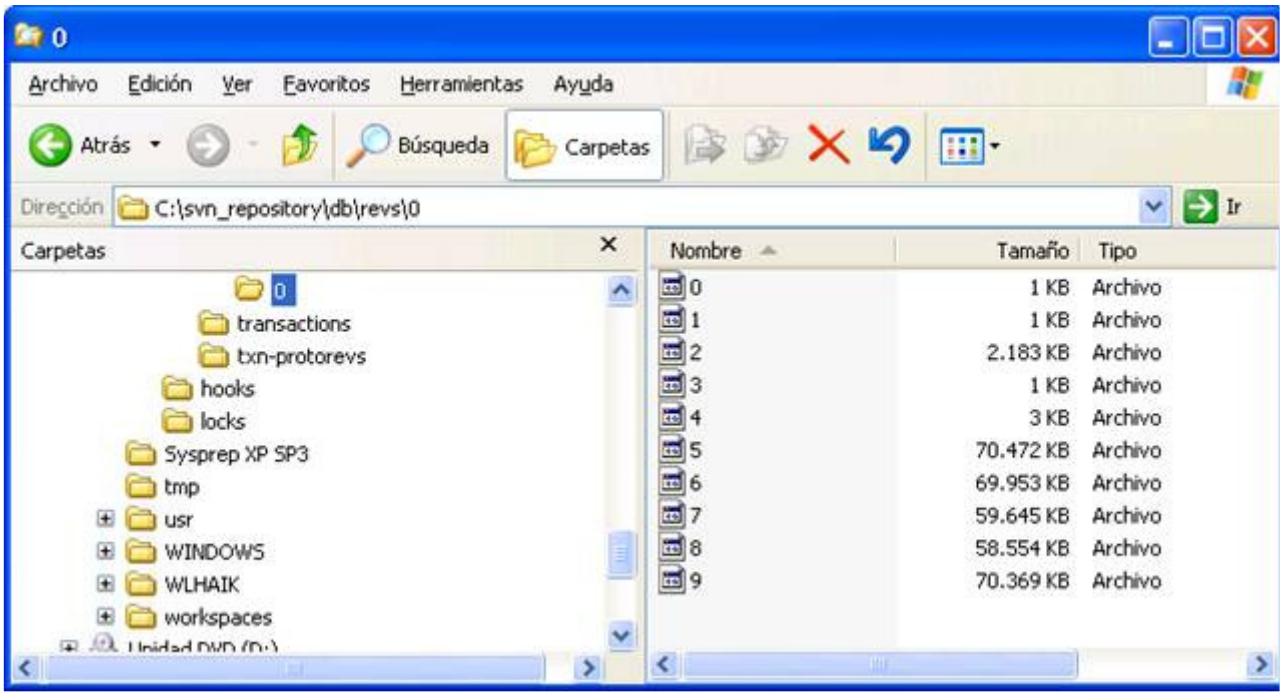
Añora voy a añadir una sola pista al archivo segundo



Vuelvo a subir al control de versiones



Y esto es lo más interesante. Ahora sólo ha transferido 25 Mbytes, que coincide con el doble de lo que suma el cambio introducido en el fichero RAR. Veamos lo que ocurre en el repositorio



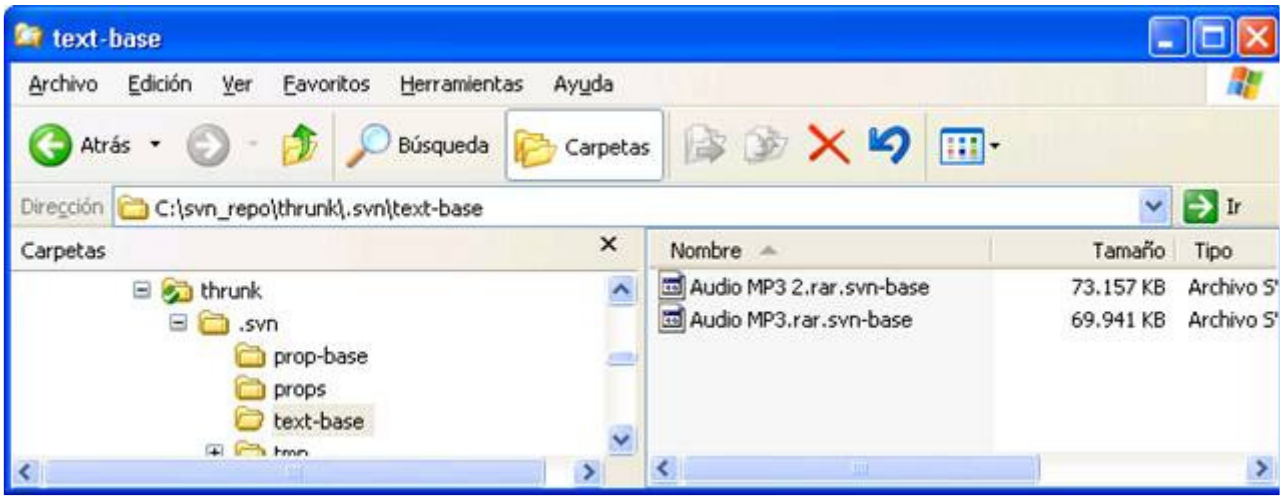
La nueva revisión ocupa 70 Mbytes, que es lo que ocupa más o menos la nueva versión del archivo de audio.

Mi conclusión es que en las subidas al control de versiones el cliente de subversion intenta subir (si no son muchas) sólo las diferencias para recrear las nuevas versiones en el repositorio, y el servidor de Subversion guarda los datos de cada fichero modificado completos, probablemente en formato comprimido. De esta forma aunque no es lo óptimo en cuanto a espacio físico en el repositorio, sí que es muy eficiente generando los ficheros para ser descargados.

Hay que puntualizar que esto depende del formato del sistema de fichero elegido para el servidor Subversion. Yo estoy comentando el formato que utiliza por defecto el servidor subversion en la versión evaluada, pero esto se puede modificar e incluso puede cambiar en otras versiones del servidor.

Además, el repositorio local mantiene una copia oculta de todos los ficheros añadidos al control de versiones, en la revisión que se obtuvo del repositorio. Esto permite generar al cliente de subversion los cambios entre lo almacenado en el repositorio y el fichero local, para enviar dichos cambios de manera eficiente al servidor.

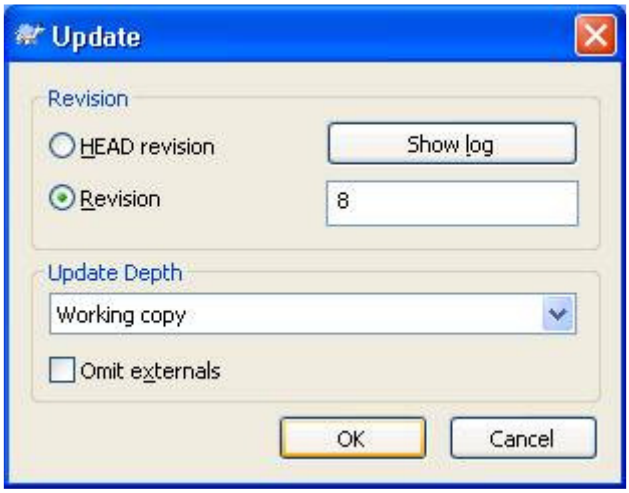
En cuanto a la copia local, nos podemos preguntar ¿cual sería actualmente el tamaño de la carpeta que mantiene la copia local del repositorio? En total tenemos dos fichero de 73 y 69 Mbytes, más una carpeta de unos 2 Mbytes de fuentes, lo que suman unos 144 Mbytes en total, aproximadamente. Si comprobamos el tamaño de la carpeta de la copia local mirando sus propiedades nos da unos 284 Mbytes, que aproximadamente corresponde con el doble de lo que ocupan los archivos de la copia local. Ya sabemos que esto se debe a la copia oculta que guarda subversion de cada archivo en la carpeta local:



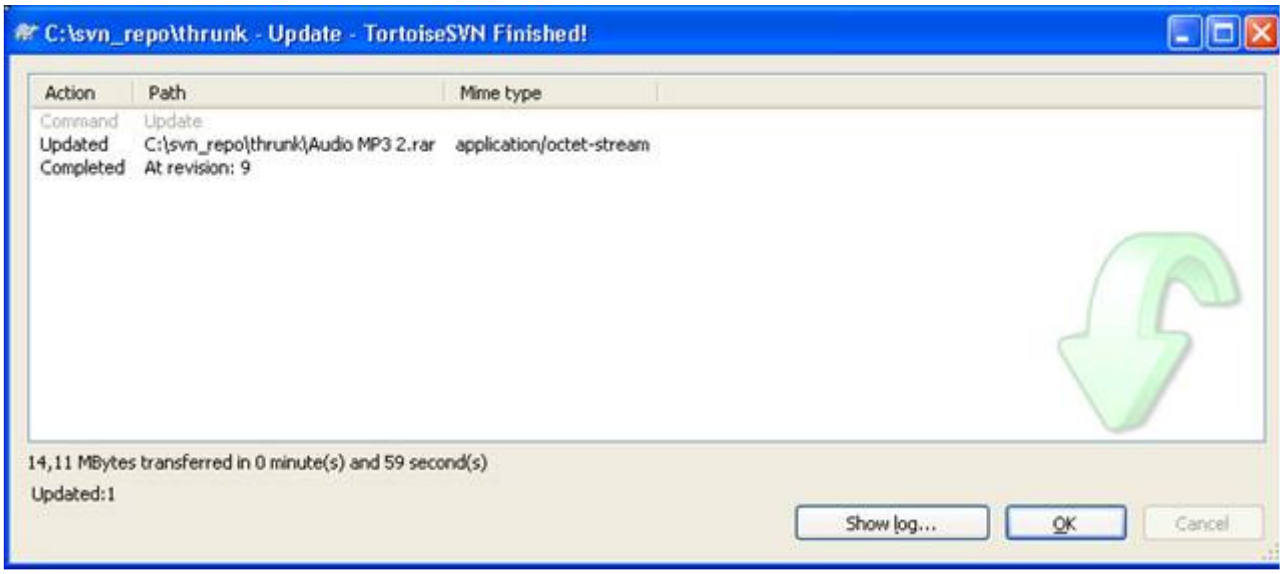
Efectivamente, el cliente de subversion guarda las copias locales dentro de la carpeta .svn/text-base con extensión svn-base

Prueba 4. Descarga parcial de archivos en las actualizaciones

Ahora vamos a ver qué ocurre en las descargas de archivos versionados
Primero seleccionamos svn_repo/thrunk y hacemos “Tortoise SVN \ Update to revision ..”



Al actualizarse la copia local el fichero de prueba que estoy usando ocupa 60,8Mbytes.
Ahora hago un “SVN update” sobre la carpeta thrunk



Lo que observamos, como ya esperamos, es que para actualizar desde la revisión 8 a la 9 el fichero "Audio MP3 2.rar" sólo ha sido necesario transferir 14.11 Mbytes, que es el tamaño de los cambios entre ambas versiones del fichero.

Conclusiones de las pruebas

Subversion nos demuestra que es un buen sistema de control de versiones de los archivos, con gran cantidad de herramientas que facilitan el mantenimiento del repositorio centralizado. Y como ventajas hemos podido observar que:

- El repositorio central reside en una base de datos basada en un potente sistema de ficheros.
- Las operaciones de creación de etiquetas, ramas y copiado de carpetas y ficheros apenas tienen coste en el sistema de ficheros.
- Cada fichero modificado es subido o descargado desde el servidor mediante los cambios parciales, lo que salvaguarda el ancho de banda de la red.
- Dentro de la base de datos, todos los ficheros correspondientes a una revisión se encapsulan en un único fichero, lo que hace que el rendimiento en espacio para carpetas repletas de ficheros sea muy alto.
- El tamaño de la copia local del repositorio suele ser del doble aproximadamente del tamaño de los ficheros existentes en el repositorio para una revisión dada, ya que se deben guardar copias ocultas de cada fichero local.

Ahora ya no tenemos ninguna excusa para no montarnos un servidor Subversion en casa, ir practicando y organizando nuestros documentos, imágenes y software.

Ejemplos de uso del control de versiones Subversion

Algunos ejemplos de uso que se me ocurren para nuestra casa y la oficina:

- Nuestros documentos y datos. Basta crearnos un repositorio para todos nuestros documentos en una unidad diferente a nuestro disco

principal de datos, por ejemplo en un disco externo, y de vez en cuando ir subiendo cambios al control de versiones. No olvidar hacer copia de seguridad del repositorio a menudo.

- Almacén de fotografías. Los repositorios subversion tienen la capacidad de añadir información adicional a los ficheros almacenados en el control de versiones, lo que normalmente se conoce como metainformación. Se puede añadir palabras clave, información geográfica, títulos, descripciones y categorías, entre otras. Mediante la API de programación del servidor Subversion se pueden crear aplicaciones que muestren el repositorio de manera avanzada, por ejemplo almacenando y mostrando imágenes de muestra (thumbnails) o la metainformación a la vez que se navega por el repositorio.
- Una biblioteca de software. A menudo debemos crearnos una biblioteca de software para todos los PCs de nuestra casa u oficina. Podemos hacer lo siguiente. Creamos el repositorio SoftLib::
    - Soft. En esta carpeta introduciremos todos los programas que necesitemos instalar en cualquier PC.
    - Soft_PC1
    - Soft_PC2, etc. Crearemos una carpeta dentro del repositorio por cada PC que deseemos instalar
    - En cada PC que vamos a instalar creamos una carpeta Soft

Múltiples repositorios en un **único servidor Subversion**

Ahora tenemos otra pregunta interesante. Con la instalación de Subversion que hemos realizado, todas las carpetas del repositorio comparten el mismo sistema de gestión de revisiones. Incluso el control de acceso a las mismas es compartido. Hay varios motivos por los que sería interesante que un mismo servidor de Subversion gestionase diferentes repositorios:

- Gestión más detallada del control del acceso y seguridad de los repositorios
- Gestión de las revisiones independiente para cada repositorio
- Facilidad en la realización de copias de seguridad parciales
- Posibilidad de crear repositorios de sólo lectura independientes para acceder a ellos por motivos históricos
- Posibilidad de llevar a servidores de menos carga los repositorios de manera independiente

Si recordamos esto para el control de versiones CVSNT era muy sencillo, pues la gestión de múltiples repositorios se hacía desde el propio panel de control de CVSNT. En el caso de Subversion con Apache, hay que hacer un poco de trabajo “manual”.

Gestión de múltiples repositorios dentro del mismo servidor

Nuestra situación inicial es un único repositorio accesible en la url http://localhost:8090/svn. Nuestra intención es crear dos repositorios separados, repositorio1 y repositorio2, accesibles desde urls diferentes, por ejemplo http://localhost:8090/svn/repositorio1 y http://localhost:8090/svn/repositorio2

Con el servidor Subversion y Apache parado realizamos las siguientes operaciones:

1. Creamos la carpeta C:\svn_repository\repositorio1

2. Creamos la carpeta C:\svn_repository\repositorio2
3. creamos el repositorio en C:\svn_repository\repositorio1. Para ello abrimos una ventana de comandos en la carpeta C:\svn_repository y ejecutamos el comando "c:\Subversion\bin\svnadmin.exe create repositorio1
4. Creamos el repositorio en C:\svn_repository\repositorio1. Para ello abrimos una ventana de comandos en la carpeta C:\svn_repository y ejecutamos el comando "c:\Subversion\bin\svnadmin.exe create repositorio2
5. Movemos el contenido anterior de la carpeta C:\svn_repository (excepto las carpetas nuevas) a C:\svn_repository\repositorio1.
6. Modificamos el fichero de configuración del Apache HTTPD

C:\Subversion\Apache2.2\conf\httpd.conf

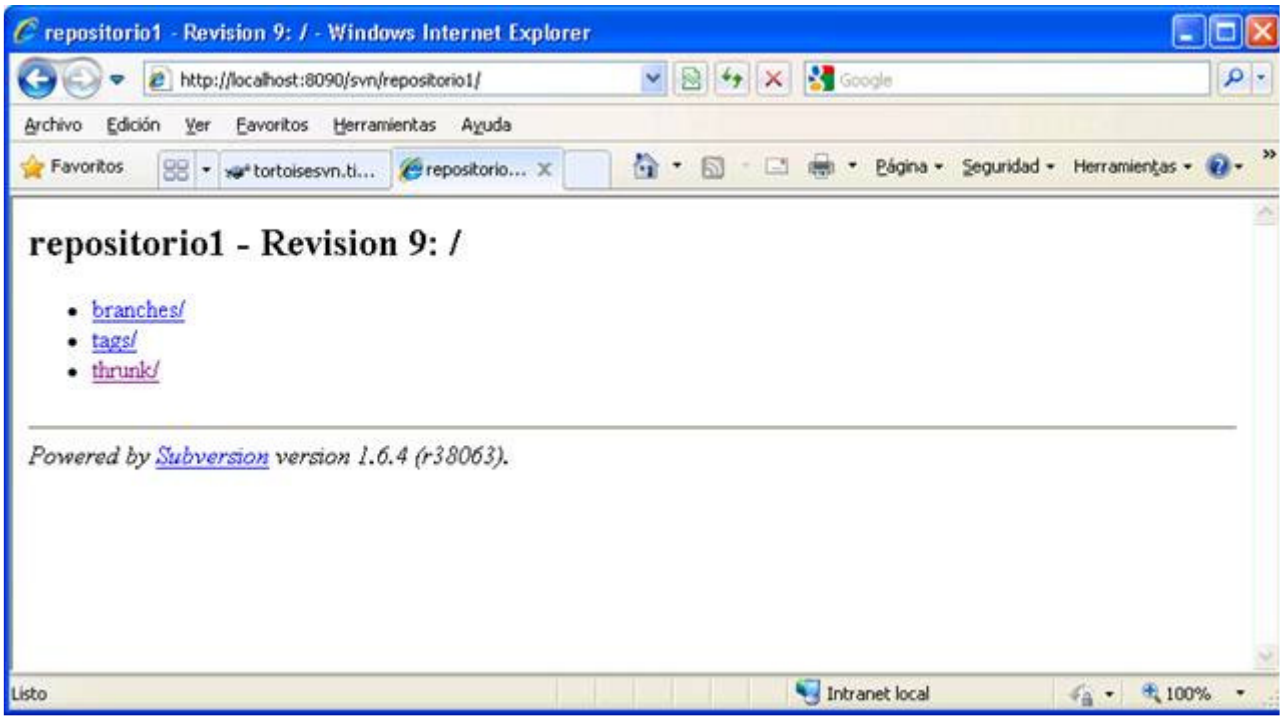
```
…

# For Subversion repository
<Location /svn/repositorio1>
   DAV svn
   SVNListParentPath on
   SVNPath /svn_repository/repositorio1
   AuthType Basic
   AuthName "Proyectos y documentos"
   AuthUserFile /Subversion/Apache2.2/passwd
   Require valid-user
</Location>

# For Subversion repository
<Location /svn/repositorio2>
   DAV svn
   SVNListParentPath on
   SVNPath /svn_repository/repositorio2
   AuthType Basic
   AuthName "Proyectos y documentos"
   AuthUserFile /Subversion/Apache2.2/passwd
   Require valid-user
</Location>
…
```

Ahora rearrancamos el servidor Subversion y el HTTPD y nos conectamos con el navegador a las url que arriba indicamos, asegurándonos primero de limpiar la caché del navegador (o usar la tecla de control al pinchar en las URLs)

Para el primer repositorio obtenemos:



Para el segundo repositorio obtenemos (recordamos que está vacío):

Conclusión

En este artículo hemos visto cómo trabaja el servidor Subversion con su repositorio. Hemos podido comprobar que el sistema de revisiones del repositorio unido a su sistema de copia interna de ficheros versionados facilita de manera sencilla la creación de ramas y etiquetas. Hemos también podido comprobar cuál es el coste real de subir un fichero al repositorio. Para trabajar con Subversion sugiero lo siguiente:

- Utiliza la operación de copiar ficheros versionados siempre que puedas, ya que no se copia el fichero internamente, sino que simplemente se genera un enlace interno al fichero copiado, al estilo de los enlaces duros de UNIX.
- Organiza tus proyectos en carpetas, y si te equivocas, simplemente copia o mueve usando las operaciones de Subversion. El repositorio es un sistema “vivo”.
- Utiliza ramas y etiquetas siempre que quieras. Las etiquetas y las ramas sólo generan un conjunto de enlaces a los ficheros y carpetas antiguos, por lo que su consumo es bajo.
- Si se te olvida etiquetar, usa el número de revisión como identificador.
- Usa múltiples repositorios para organizar la información. Incluso dispones de herramientas que te permiten extraer partes de repositorios a nuevos repositorios.
- Monta los repositorios históricos como sólo lectura. Así siempre tendrás accesible la información.
- Educa a todos los usuarios del grupo u organización a usar el repositorio. En desarrollo aplica la máxima: “Lo que no está en el repositorio no existe”.
- Para los desarrolladores, acostúmbralos a subir todos los días los ficheros fuentes y que comprueben que “compilan” al menos. Antes de empezar un nuevo desarrollo no trivial se debería crear

una rama del mismo, pues así se puede trabajar con más libertad, y como hemos visto el coste de esta operación es muy pequeño y fácil de realizar por cualquier usuario.

Espero que con esta serie de artículos más personas se animen a montar sus propios servidores Subversion en su casa o en su organización.

Anímate y coméntanos lo que pienses sobre este **TUTORIAL:**

Puedes opinar o comentar cualquier sugerencia que quieras comunicarnos sobre este tutorial; con tu ayuda, podemos ofrecerte un mejor servicio.

Enviar comentario

(Sólo para usuarios registrados)

» **Registrate** y accede a esta y otras ventajas «

COMENTARIOS