FACULTADE DE INFORMÁTICA
Departamento de Computación
Tecnología de la Programación 2007-2008

Práctica 1: Subversion

Introducción

El objetivo de esta práctica es aprender a manejar Subversion y afianzar los conceptos comentados en el seminario de la semana pasada.

Cada grupo de prácticas dispone de su propio repositorio en la URL https://www.madsgroup.org/docencia/svn/tp0708/gXXXp_svn donde gXXX es el identificador asignado a dicho grupo.

Para acceder al repositorio se utilizará *svn*, el cliente de Subversion por línea de comandos.

- *svn help* describe el uso del cliente y muestra los subcomandos disponibles.
- s*vn help <subcomando>* describe la sintaxis, opciones y comportamiento del subcomando en cuestión.

Nota: la práctica se entregará al final de las horas de prácticas (la entrega serán los ficheros subidos al propio repositorio al final de la clase).

1. Creación del repositorio

Los repositorios de prácticas tendrán la siguiente estructura de directorios:

/trunk

/branches

/tags

El directorio *trunk* contiene la línea actual de desarrollo, es decir, los ficheros de prácticas con los que se está trabajando. El directorio *tags* contiene versiones concretas de la práctica. Por tanto, tras crear un *tag, é*ste no debe modificarse. Así mismo con el directorio branches que almacenará las distintas ramas del proyecto si algún día existen.

La primera tarea a realizar consistirá en crear los directorios *trunk, tags y branches* en el repositorio. Una vez creados dichos directorios, se listará el contenido del repositorio para comprobar la existencia de ambos directorios.

2. Añadir ficheros

A continuación, añadiremos un programa simple en java que lee un fichero especificado por línea de comandos, y lo imprime por pantalla. Este programa estará en un nuevo directorio P1 dentro del trunk, junto con todos los ficheros que se generaran a lo largo de esta práctica.

Además, se proveerá de un makefile para compilar.

```
**************************************************************
```
LeerFichero.java

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;


public class LeerFichero {

   public static void main(String args[]){
      if(args.length < 1) help();

      BufferedReader br;
      try {
         br = (new BufferedReader(new FileReader(new File(args[0]))));
      } catch (FileNotFoundException e) {
         help();
      }
      String line;
      try {
         while((line = br.readLine()) != null){
            System.out.println(line);
         }
      } catch (IOException e) {
         e.printStackTrace();
         System.exit(-1);
      }
   }

   public static void help(){
      System.out.println("uso: java LeerFichero FICHERO");
      System.exit(0);
   }
}
**************************************************************
```
Makefile

```
all:LeerFichero.class
LeerFichero.class:LeerFichero.java
      javac LeerFichero.java
clean:
      rm LeerFichero.class
**************************************************************
```

NOTA: Los comandos en ficheros Makefile deben ir separados por tabuladores. Para compilar se ejecuta el comando make y debería hacer una compilación correctamente.

Una vez creados, serán incorporados al repositorio en el directorio *trunk* incluyendo el comentario "Versión inicial". Se listará el contenido del directorio *trunk* del repositorio para comprobar que los ficheros han sido copiados.

3. Descargar el repositorio

A pesar de haber copiado los ficheros al repositorio, las copias locales no se encuentran todavía bajo control de versiones. Para ello, se descargará la última versión del directorio *trunk* del repositorio. De

esta forma se creará una “copia de trabajo” en la cuenta del usuario.

4. Realizar cambios locales

Al intentar compilar el programa se producirá un error. La cuarta tarea consistirá en corregir el programa y subirlo de nuevo al repositorio indicando la corrección realizada mediante un comentario. Se comprobarán las diferencias existentes entre la revisión inicial y la versión corregida.

5. Añadir nuevos ficheros al repositorio (un usuario)

Añadiremos nuevos ficheros al repositorio.

Para ello, modificaremos el programa que ya tenemos transformándolo en un programa que en vez de leer la ruta de fichero desde línea de comandos (args[0]) leerá la que está especificada en un fichero .properties. Y, ¿qué es un fichero .properties?

El API de Java nos proporciona la clase Properties, la cual permite de forma sencilla manejar ficheros de configuración (habitualmente con la extensión .properties). Estos ficheros constan de un conjunto de pares CLAVE=VALOR (cada uno en una linea) y comentarios (líneas que comienzan por #). Para más información, ver http://java.sun.com/j2se/1.5.0/docs/api/java/util/Properties.html

Atención: Antes de modificar LeerFichero.java crear un tag con el nombre "version 1". (Como crear un TAG.... pistas al final)

En nuestro caso, especificaremos la ruta del fichero con la propiedad con ingenioso nombre, "ruta" en el fichero con no menos ingenioso nombre "leer.properties" que colocaremos en el directorio "configuracion" todo dentro del directorio P1.

Los ficheros que queremos que lea el programa los colocaremos en el directorio "ficheros", y creamos un archivo de texto de ejemplo (al que le llamamos chorrada1.txt, por ejemplo).

El nuevo programa quedaría como sigue:

****

LeerFichero.java

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.FileInputStream;
import java.util.Properties;


public class LeerFichero {

    public static void main(String args[]){

        BufferedReader br = null;
        String CONF = "configuracion/leer.properties";
        String file = null;
        FileInputStream fis;

        try {
```

```
            Properties propiedades;
                fis = new FileInputStream(CONF);
                propiedades = new Properties();
                propiedades.load(fis);
                fis.close();
                file = propiedades.getProperty("ruta");
            } catch (Exception e1) {
                System.out.println("Fichero de configuracion "+CONF+" no encontrado!");
                System.exit(-1);
            }
        try {
            br = (new BufferedReader(new FileReader(new File(file))));
        } catch (FileNotFoundException e) {
            System.out.println("Fichero "+file+" no encontrado!");
            System.exit(-1);
        }

        String line;
        try {
            while((line = br.readLine()) != null){
                System.out.println(line);
            }
        } catch (IOException e) {
            e.printStackTrace();
            System.exit(-1);
        }
    }
}
```

****

configuracion/leer.properties

```
ruta=ficheros/chorrada1.txt
```

****

Resumen para los despistados

- Tagear el trabajo actual como "version-1"
- Modificar el fichero LeerFichero.java
- Crear el directorio "configuracion" y meter ahí el fichero "leer.properties"
- Crear el directorio "ficheros" y meter ahi el fichero "chorrada1.txt"
- Añadir los ficheros nuevos al repositorio
- Comprobar el estado de los ficheros y de los cambios realizados
- Guardar los cambios en el repositorio

Y como tareas a mayores:

- Tagear la nueva versión como "version-2"
- Comprobar las diferentes versiones/revisiones del fichero LeerFichero.java
- Recuperar la versión anterior del fichero LeerFichero.java
- Revisar el log de operaciones

6. Modificación de ficheros sin conflicto (dos usuarios)

Para los últimos puntos de la práctica, cada uno de los miembros del grupo trabajará de forma independiente con el repositorio desde su propia cuenta. Es posible simular estos ejercicios con un sólo usuario duplicando la copia de trabajo en un directorio diferente.

Cada alumno se descargará la última revisión del repositorio en su cuenta. Cada uno generará dos archivos de texto con nombres diferentes, que se copiarán al directorio "ficheros".

Una vez que ambos alumnos han guardado los cambios en el repositorio, actualizarán su copia de trabajo y comprobarán que tienen el fichero creados por su compañero. Además, se comprobará el log de cambios.

7. Modificación de ficheros con conflicto (dos usuarios)

Ahora vamos a liarla, chicos.

Cada uno de los alumnos modificará su copia del fichero a leer (chorrada.txt).

El primer alumno (PACO) en guardar los cambios en el repositorio podrá hacerlo sin problemas mientras que el segundo (PEPA) se encontrará con un conflicto ya que la copia del repositorio que ha descargado del repositorio y ha modificado, acaba de ser modificada en el repositorio por su compañero.

Para resolver el conflicto en el fichero, PEPA (el segundo alumno) tendrá que actualizar su copia de trabajo. Tendrá ahora 4 versiones del fichero chorrada.txt:

- *chorrada.x*: fichero con las modificaciones (del primer y del segundo alumno) de PACO y PEPA marcadas de la siguiente forma:

<<<<<<< .mine

<contenido añadido por PEPA (del segundo alumno)>

=======

<contenido de la nueva revisión guardada por PACO (el primer alumno)>

>>>>>>> .rNUEVA

- *chorrada.txt.mine*: fichero hola.c de PEPA (del segundo alumno)

- *chorrada.txt.rANTERIOR*: fichero tal y como se lo había descargado PEPA (el alumno 2)

- *chorrada.txt.rNUEVA*: fichero tras haber guardado los cambios PACO (el primer alumno)

PEPA (El segundo alumno) deberá resolver los conflictos editando los ficheros y modificando lo que considere conveniente. Tras ello, PEPA (el segundo alumno) deberá indicar al repositorio que ha resuelto el conflicto y guardará los cambios en el repositorio documentando los cambios que ha introducido.

8. Ejemplo de Ramas y tags

Para la creación de ramas y tags en un repositorio Subversión se utiliza el comando svn copy, para crear un tag (marca) que nos identifique una versión de nuestro proyecto simplemente deberémos ejecutar el comando svn copy del trunk al directorio tags con una notificación de versión.

Por ejemplo

$svn copy trunk tags/version-1

Si decido no enviar ningún cambio al subdirectorio tags/version-1, se le llama una **etiqueta (tag)**. De lo contrario, es una **rama (branch)**. Al disponer de una etiqueta es posible descargar directamente /tags/vesrion-1 y obtener el código fuente de una versión en concreto sin tener que recordad el número de revisión de la misma.

Las ramas son utilies cuando se realizan cambios sobre versiones anteriores, o se quiere abrir una nueva rama de desarrollo experimental sin afectar a la línea base del código fuente, los desarrolladores de subversión recopienda disponer de un directorio branches junto con el trunk y el tags para almacenar las ramas.

Más información en: http://svnbook.red-bean.com/nightly/es/svn-ch-4.html