Java Application: CompareFolders

Written by: Keith Fenske, http://www.psc-consulting.ca/fenske/
First version: Friday, 2 May 2008
Document revised: Saturday, 13 February 2010

Description

CompareFolders is a Java 1.4 application to compare two folders to determine if all files and subfolders are identical. The folders may be on the same computer, on the local network, or they may be represented by checksum files. Files or subfolders that are not the same are reported to the user. The intention is to decide if two different distribution folders have the same contents. You can:

1. Compare two folders. One or both folders may be represented by checksum files.
2. Create a checksum file for a folder. This is the most efficient way of remembering the contents of a folder for later comparison.
3. Update a checksum by assuming that files have not changed if they have the same name, date, and size. This is much faster than creating a new checksum for large collections on a local disk drive. However, it is not recommended for files copied over a network or downloaded from the internet.
4. Find duplicate files with the same MD5 or SHA1 checksum.

Checksum files are used when the original files or folders are not available. A checksum file is a plain text file in XML (Extensible Markup Language) format with the name, size, and checksums for each file. Checksums are small hexadecimal "signatures" for testing whether or not files have been copied correctly, such as over a network. One person sends a file along with the checksum computed on the original computer. A second person calculates a similar checksum for the received file, and if the two checksums agree, then the received file is assumed to be correct. This CompareFolders application supports CRC32, MD5, and SHA1 checksums. It is extremely unlikely that two files will have the same MD5 or the same SHA1 checksum and still be different (which can't be said about CRC32). For documentation on XML, see:

> Extensible Markup Language (XML)
> http://en.wikipedia.org/wiki/XML
> http://www.xml.com/
> http://www.w3.org/XML/

The XML output produced by this program can be read by most XML applications, such as those on many internet web sites. See the FileChecksum Java application for an easy way to generate or test checksums for a single file. See the FindDupFiles Java application to look for duplicate files based on MD5 checksums. CompareFolders differs from UNIX “md5sum” and “sha1sum” and Windows “MD5summer” in that it supports multiple checksums and uses XML format for storing checksums in text files. Other programs for exact file comparisons (byte-by-byte) are “comp” on DOS/Windows and “cmp” on UNIX; “WinDiff” on Windows will compare folders and subfolders.

GNU General Public License (GPL)

Installation

You must have the Java run-time environment (JRE) installed on your computer. CompareFolders was developed with Java 1.4 and should run on later versions. It may also run on earlier versions, but this has not been tested. For Macintosh computers, the version of Java is determined by your version of MacOS. For Windows, Linux, and Solaris, you can download the JRE from Sun Microsystems:

Sun Java
- JRE for end users: http://www.java.com/getjava/
- SDK for programmers: http://developers.sun.com/downloads/
- IDE for programmers: http://www.netbeans.org/

Once Java is installed, you need to put the program files for CompareFolders into a folder (directory) on your hard drive. The name of the folder and the location are your choice, except it is easier if the name does not include spaces. Assume that files will go into a C:\JAVA folder. Then create the folder and unpack the Java *.class files into this folder (if you received the program as a ZIP file). The files look something like this:

```
CompareFolders3.class  (52 KB, executable program)
CompareFolders3.doc  (39 KB, this documentation in Microsoft Word format)
CompareFolders3.gif  (14 KB, sample program image)
CompareFolders3.ico  (4 KB, icon for Windows)
```

```
CompareFolders3.jar  (28 KB, archive file with same class files inside)
CompareFolders3.java  (223 KB, source code)
CompareFolders3.manifest  (1 KB, main class manifest for archive file)
CompareFolders3.pdf  (81 KB, this documentation in Adobe Acrobat format)
CompareFolders3File.class  (2 KB, helper class for main program)
CompareFolders3User.class  (1 KB)
GnuPublicLicense3.txt  (35 KB, legal notice)
RunJavaPrograms.pdf  (88 KB, more notes about running Java)
```

To run the program on Windows, start a DOS command prompt, which is Start button, Programs, Accessories, Command Prompt on Windows 2000/XP. Change to the folder with the program files and run the program with a "java" command:

```
c:
cd  \java
java  CompareFolders3
```

The program name "CompareFolders3" must appear exactly as shown; uppercase and lowercase letters are different in Java names. Some systems (Macintosh) will run a main "class" file by clicking on the class file name while viewing a directory in the file browser (Mac Finder). Many systems will run a "jar" file by clicking (or double clicking) on the jar file name (Windows Explorer). The command line is the only guaranteed way of running a Java program. Should you find this program to be popular, you can create a Start menu item or desktop shortcut on Windows 2000/XP with a target of "java.exe CompareFolders3" starting in the "c:\java" folder.

One complication may arise when trying to run this program. Java looks for an environment variable called CLASSPATH. If it finds this variable, then that is a list of folders where it looks for *.class files. It won't look anywhere else, not even in the current directory, unless the path contains "." as one of the choices. The symptom is an error message that says:

```
Exception in thread "main" java.lang.NoClassDefFoundError: CompareFolders3
```

To find out if your system has a CLASSPATH variable defined, type the following command in a DOS window:

```
set  CLASSPATH
```

To temporarily change the CLASSPATH variable to the current directory, use the following command line:

```
java  -cp  .  CompareFolders3
```

To permanently change the CLASSPATH, you must find where it is being set. This may be in an old AUTOEXEC.* file in the root directory of your system disk (usually the C:\ folder), or it may be in Control Panel, System, Advanced, Environment Variables on Windows 2000/XP.

Removal or Uninstall

To remove this program from your computer, delete the installation files listed above. If the folder that contained the files is now empty, you may also delete the folder ... if you created the folder, of course, not the system. If you created desktop shortcuts or Start menu items, then delete those too. There are no hidden configuration or preference files, and no information is stored in the Windows system registry. You don't need an "uninstall" program.

Graphical Versus Console Application

The Java command line may contain options or file and folder names. If no file or folder names are given on the command line, then this program runs as a graphical or "GUI" application with the usual dialog boxes and windows. See the "-?" option for a help summary:

```
java CompareFolders3 -?
```

The command line has more options than are visible in the graphical interface. An option such as -u14 or -u16 is recommended because the default Java font is too small. If file or folder names are given on the command line, then this program runs as a console application without a graphical interface. A generated report is written on standard output, and may be redirected with the ">" or "1>" operators. (Standard error may be redirected with the "2>" operator.) An example command line is:

```
java CompareFolders3 -s d:\fonts d:\temp\checksum.txt >errors.txt
```

The console application will return an exit status of 1 for success, –1 for failure, and 0 for unknown. The graphical interface can be very slow when the output text area gets too big, which will happen if thousands of files are reported. For both with over 20,000 files, you may need to set the "-Xmx" option on the "java" command to allocate more memory:

```
java -Xmx150M CompareFolders3
```

would give the program a maximum of 150 megabytes (MB) of memory for temporary data. The default is around -Xmx60M or 60 megabytes for Java 1.4 on Windows 2000.

Buffer size is one option that can significantly affect speed in some situations. Default sizes are nearly optimal when files are on hard disk drives: 64 KB for checksums and 4 MB for comparisons. (A larger comparison size avoids thrashing the disk drive.) Network shares and

CD/DVD drives are faster with smaller buffer sizes around 1 KB, and the best size may be slightly larger depending upon the speed of your devices. This is not something that a Java program can recognize and compensate for. If performance is slower than you would expect, try changing the “-b” option on the command line, and be very careful about judging the results. Buffer sizes are almost always a power of two: 1 KB, 2 KB, 4 KB, 8 KB, etc. By the way, comparing two folders on the same CD/DVD disc is not recommended, due to the long latency of “seek” operations on essentially sequential media.

Restrictions and Limitations

The XML parser used by this program to read checksum files is very simple and can not handle the full XML language, as would be supported in optional Java 1.5 (5.0) and later packages. Only simple tags (elements) without attributes are recognized. Everything else is either ignored or will generate an error. The parser is really only smart enough to read back its own output files.

File modification time stamps in checksum files can vary depending upon the current time zone and rules for daylight saving time (DST). This may cause the “update checksum” action to recompute checksums when in fact nothing has changed. Refreshing checksums occasionally is a good idea, but it would be better if the date and time had an invariant form. The problem has been confirmed with Java 1.4 through 6 on Windows 2000/XP and for FAT32 volumes. NTFS volumes are not affected. Both Java and Windows are adjusting for DST: Windows 2000/XP uses the current DST offset, and Java tries to be historically correct. Java, of course, gets its information from the underlying operating system, after any changes that the OS makes.

Please avoid slashes (/) or backslashes (\) in file and folder names. These are illegal on Windows but accepted on Macintosh computers. The Java run-time may incorrectly parse these as additional file or folder separators, or may substitute them with other characters that obviously won’t match the original file names. Slashes have always had a special meaning on UNIX-based operating systems, including Linux and Mac OS X. For maximum compatibility, you should avoid all of the following characters: " * / : < > ? \ | and don’t start a name with “.” or a space, and don’t end a name with a space.

---

file: CompareFolders3.doc 2010-02-13