# Move / Copy File Operations in Java

Asked 16 years, 1 month ago  Modified 5 years, 3 months ago

Viewed 112k times

71

Is there a standard Java library that handles common file operations such as moving/copying files/folders?

java file copy move

Share

Improve this question

Follow

edited Jun 7, 2009 at 14:23

Jonik

81.7k • 76 • 269 • 379

asked Nov 18, 2008 at 23:34

MSumulong

1,061 • 2 • 12 • 22

## 7 Answers

Sorted by: Highest score (default)

73

Here's how to do this with `java.nio` operations:

```
public static void copyFile(File sourceFile, File dest
{
    if(!destFile.exists()) {
        destFile.createNewFile();
    }

    FileChannel source = null;
    FileChannel destination = null;
```

```
    try {
        source = new FileInputStream(sourceFile).getCh
        destination = new FileOutputStream(destFile).g

        // previous code: destination.transferFrom(sou
        // to avoid infinite loops, should be:
        long count = 0;
        long size = source.size();
        while((count += destination.transferFrom(sourc
<size);
    }
    finally {
        if(source != null) {
            source.close();
        }
        if(destination != null) {
            destination.close();
        }
    }
}
```

Share Improve this answer

Follow

edited Apr 30, 2012 at 16:35



pcarter

**1,586** ● 14 ● 21

answered May 26, 2009 at 7:28

Rigo Vides

**1,364** ● 13 ● 17

7 if the file exists will the content be append or overwritten?
– Janusz Jul 7, 2009 at 19:12

@Rigo This moves only files, i'm not able to move directories
– Arasu Nov 23, 2011 at 10:57

11 It's worth noting that Java 7 has simpler copy/move methods.
– Tharwen May 25, 2012 at 15:49

2 Any reason why `File.renameTo` isn't as good as `Files.move`? – Erik Kaplun Mar 5, 2014 at 19:42

4 @Erik Allik: If the operation fails, `Files.move` will tell you *why* instead of simply returning `false`. And `Files.move` can handle moves that are not simple renames, i.e. need a copy-and-delete. – Holger Oct 7, 2014 at 14:41

41

Not yet, but the New NIO (JSR 203) will have support for these common operations.

In the meantime, there are a few things to keep in mind.

File.renameTo generally works only on the same file system volume. I think of this as the equivalent to a "mv" command. Use it if you can, but for general copy and move support, you'll need to have a fallback.

When a rename doesn't work you will need to actually copy the file (deleting the original with File.delete if it's a "move" operation). To do this with the greatest efficiency, use the FileChannel.transferTo or FileChannel.transferFrom methods. The implementation is platform specific, but in general, when copying from one file to another, implementations avoid transporting data back and forth between kernel and user space, yielding a big boost in efficiency.

Share Improve this answer Follow

edited Nov 18, 2008 at 23:43

answered Nov 18, 2008 at 23:38

erickson

**269k** ● 59 ● 401 ● 497

---

17

Check out: http://commons.apache.org/io/

It has copy, and as stated the JDK already has move.

Don't implement your own copy method. There are so many floating out there...

Share Improve this answer Follow

edited Nov 18, 2008 at 23:59

answered Nov 18, 2008 at 23:41

Pyrolistical

**28k** ● 21 ● 84 ● 109

Commons IO has limitations with respect to the size of files it can copy. For a general-purpose solution, a more robust implementation would be expected. – erickson Nov 19, 2008 at 0:14

3 Implementing one's own copy method is trivial and means you won't be dependent on an entire library. *Do* implement your own – oxbow_lakes Nov 19, 2008 at 8:57

17 Copy method is far from trivial. You can easily make a correct one that doesn't perform using Streams, and fast but incorrect one using NIO. Never implement your own utilities when there are quality libraries out there. – Pyrolistical Nov 21, 2008 at 17:59

3 A copy method *is* non-trivial, and Apache Commons can't handle a common use case: information too large for main

memory. A library intended for managing mass storage should have bounds on its memory-consumption, which Apache Commons move method lacks. – erickson Dec 9, 2009 at 18:47

@Pyrolistical `Never implement your own utilities when there are quality libraries out there.` Uhh suure, if you never have to worry about licensing. – arkon May 23, 2012 at 3:44

10

Previous answers seem to be outdated.

Java's File.renameTo() is probably the easiest solution for API 7, and seems to work fine. Be carefull IT DOES NOT THROW EXCEPTIONS, but returns true/false!!!

Note that there seem to be problems with it in previous versions (same as NIO).

If you need to use a previous version, check here.

Here's an example for API7:

```
File f1= new File("C:\\Users\\.....\\foo");
File f2= new File("C:\\Users\\......\\foo.old");
System.err.println("Result of move:"+f1.renameTo(f2));
```

Alternatively:

```
System.err.println("Move:" +f1.toURI() +"--->>>>"+f2.t
Path b1=Files.move(f1.toPath(),  f2.toPath(), Standard
,StandardCopyOption.REPLACE_EXISTING ););
System.err.println("Move: RETURNS:"+b1);
```

Share Improve this answer

Follow

edited Sep 10, 2019 at 10:50

Rakesh

4,242 ● 2 ● 20 ● 31

answered Jun 6, 2012 at 13:02

ntg

14k ● 9 ● 80 ● 103

3 If you are getting "The process cannot access the file because it is being used by another process."The process cannot access the file because it is being used by another process." exception on the second piece of code, remember to close the file before moving it..... – ntg Jun 6, 2012 at 14:54

2 There are other unexpected situations in which it fails, e.g. on linux if you have two different filesystems mounted under /mnt/a /mnt/b, you cannot rename a file /mnt/a/file1 to /mnt/b/file2, since it actually is a move operation, File.renameTo would fail in this case. – xask Jan 9, 2013 at 9:00

1 This is the best solution. Just use Files.move() if you are concerned about the rename operation failing. – xtian May 25, 2014 at 16:52

8

Google's Guava library also has these:

http://guava-libraries.googlecode.com/svn/trunk/javadoc/com/google/common/io/Files.html

answered Jul 5, 2010 at 16:14

Andrew McKinlay
2,611 ● 1 ● 26 ● 27



Try to use org.apache.commons.io.FileUtils (General file manipulation utilities). Facilities are provided in the following methods:

> (1) FileUtils.moveDirectory(File srcDir, File destDir) => Moves a directory.
>
> (2) FileUtils.moveDirectoryToDirectory(File src, File destDir, boolean createDestDir) => Moves a directory to another directory.
>
> (3) FileUtils.moveFile(File srcFile, File destFile) => Moves a file.
>
> (4) FileUtils.moveFileToDirectory(File srcFile, File destDir, boolean createDestDir) => Moves a file to a directory.
>
> (5) FileUtils.moveToDirectory(File src, File destDir, boolean createDestDir) => Moves a file or directory to the destination directory.

It's simple, easy and fast.



edited Oct 21, 2013 at 20:24

Leigh
28.9k ● 10 ● 57 ● 108

answered Feb 1, 2013 at 12:54

Dellanio

309 ● 3 ● 4

Are these functions atomic? – Sumit Feb 9, 2015 at 4:33

0

Interesting observation: Tried to copy the same file via various java classes and printed time in nano seconds.

Duration using FileOutputStream byte stream: 4 965 078

Duration using BufferedOutputStream: 1 237 206

Duration using (character text Reader: 2 858 875

Duration using BufferedReader(Buffered character text stream: 1 998 005

Duration using (Files NIO copy): 18 351 115

when using Files Nio copy option it took almost 18 times longer!!! Nio is the slowest option to copy files and BufferedOutputStream looks like the fastest. I used the same simple text file for each class.

Share Improve this answer

Follow

answered Jan 9, 2017 at 9:43

Elena

1