原创 JDK1.7 Paths,Files类实现文件夹的复制与删除

2016-01-13 14:31:48 尚云峰111 阅读数 2133 更多

JDK1.7 Paths,Files类实现文件夹的复制与删除

```
public static void copyFolder(String srcFolder, String destFolder)
        throws IOException {
    long startTime = System.currentTimeMillis();
    final Path srcPath = Paths.get(srcFolder);
    // 这里多创建一级，就解决了没有外壳的问题
    final Path destPath = Paths.get(destFolder, srcPath.toFile().getName());
    // 检查源文件夹是否存在
    if (Files.notExists(srcPath)) {
        System.err.println("源文件夹不存在");
        System.exit(1);
    }
    // 如果目标目录不存在，则创建
    if (Files.notExists(destPath)) {
        Files.createDirectories(destPath);
    }
// 这里是官方例子的开头，可能是针对大文件处理设置的参数
// Files.walkFileTree(srcPath,     EnumSet.of(FileVisitOption.FOLLOW_LINKS),
// Integer.MAX_VALUE, new SimpleFileVisitor<Path>() {}
//简化后的开头
    Files.walkFileTree(srcPath, new SimpleFileVisitor<Path>() {
        // 官方还调用了专门的文件夹处理，这里没使用
        // public FileVisitResult preVisitDirectory(Path dir,
        // BasicFileAttributes attrs) throws IOException {return null;}
        @Override
        // 文件处理，将文件夹也一并处理，简洁些
        public FileVisitResult visitFile(Path file,
            BasicFileAttributes attrs) throws IOException {
        Path dest = destPath.resolve(srcPath.relativize(file));
        // 如果说父路径不存在，则创建
        if (Files.notExists(dest.getParent())) {
            Files.createDirectories(dest.getParent());
        }
        Files.copy(file, dest);
        return FileVisitResult.CONTINUE;
        }
    });
    long endTime = System.currentTimeMillis();
    System.out.println("复制成功!耗时：" + (endTime - startTime) + "ms");
    }

    // 删除文件夹
    public static void deleteFolder(String Foleder) throws IOException {
    Path start = Paths.get(Foleder);
```

```
    if (Files.notExists(start)) {
        throw new IOException("文件夹不存在！");
    }

    Files.walkFileTree(start, new SimpleFileVisitor<Path>() {
        @Override //构成了一个内部类
        //处理文件
        public FileVisitResult visitFile(Path file,BasicFileAttributes attrs) throws IOException {
        Files.delete(file);
        return FileVisitResult.CONTINUE;
        }

        @Override
        //再处理目录
        public FileVisitResult postVisitDirectory(Path dir, IOException e)
            throws IOException {
        if (e == null) {
            Files.delete(dir);
            return FileVisitResult.CONTINUE;
        } else {
            throw e;
        }
        }
    });
    System.out.println("删除成功！");
    }

    public static void main(String[] args) throws IOException {
//copyFolder("C:\\Users\\Administrator\\Desktop\\111", "D:\\压缩\\1级\\2级");// 419ms,378ms,429ms....
deleteFolder("C:\\Users\\Administrator\\Desktop\\111");}
```

附：jdk1.6的处理方法

```
/*
 * 需求： 删除指定目录中的所有文件及文件夹，将输出语句改成删除语句即可
 *    删除时，不进入回收站
 * 思路一：比较简洁，理解上有点难
 *   a.判断是否是文件夹
 *   b.是的话，获取file.listFiles()
 *   c.判断File数组是否为空
 *   d.不为空的情况下，遍历数组
 *   e.数组里面的每一个文件对再调用自己的方法
 *
 *   判断结果：
 *     不加else，就会返回所有的文件及文件夹
 *  1，不是文件的直接删除
 *  2，空目录的跳过遍历，也是直接删除
 *
 * 思路2：文件与目录各自判断对应删除
 *    a. 首先获取srcFile下的所有文件对象
 *    b. 判断File数组是否为空
 *    c. 是的话遍历File数组
 *    d. 判断是否为文件夹
 *      是：将File数组元素继续调用方法遍历
```

```
 *     否：删除文件
 *    e.跳出for后，就是空文件夹,于是删除目录
 *
 *比较：两者的删除文件的顺序是一样的，思路2比较清晰，先判断是否是文件，再删除
 *    思路1比较简洁 ，用一个 SrcFile.isDirectory() 判断就删除了文件及文件夹
 *    因为它巧妙的利用了跳出for后，就是空文件夹
 */
public class FileDemo2_1 {
    //思路1实现 需求：删除所有文件（包括文件夹）
    public static void printDelete1(File SrcFile) {
    if (SrcFile.isDirectory()) {
        File files[] = SrcFile.listFiles();
        if (files != null) {
        for (File f : files) {
            printDelete1(f);
        }
        }
    }
    //删除所有文件及文件夹.不进入回收站
    System.out.println("删除："+SrcFile.getName()+"---"+SrcFile.delete());;
    }

    //思路2实现 需求：删除所有文件（包括文件夹）
    public static void printDelete2(File SrcFile) {
    File files[]=SrcFile.listFiles();
    //为什么判断files是否为空，因为有些受保护的文件无法删除，于是返回null
    if (files!=null) {
        for(File file:files){
        if (file.isDirectory()) {
            printDelete2(file);
        }
        else{
           System.out.println("删除文件："+file.getName() + "---"+file.delete());
        }
        }
        //跳出for的肯定是最后一级的目录，于是再删除目录
        //即对于每一级目录，先删除里面的文件，在删除空文件夹
        System.out.println("删除目录："+SrcFile.getName() + "---"+SrcFile.delete());
    }
    }
    public static void main(String[] args) {
// printDelete1(new File("D:\\Program Files删除"));
    printDelete2(new File("D:\\Program Files删除"));
    }
}
```