链滴

commons-io使用笔记

作者：tianma630

原文链接：https://ld246.com/article/1377225746285

来源网站：链滴

　　commons-io是一款处理io流的工具，封装了很多处理io流和文件的方法，可以大大简化我处理io流和操作文件的代码。从common-io的官方使用文档可以看出，它主要分为工具类、尾端类行迭代器、文件过滤器、文件比较器和扩展流。
一、工具类
工具类包括FileUtils、IOUtils、FilenameUtils和FileSystemUtils，前三者的方法并没有多大的别，只是操作的对象不同，故名思议：FileUtils主要操作File类，IOUtils主要操作IO流，FilenameUtil则是操作文件名，FileSystemUtils包含了一些JDK没有提供的用于访问文件系统的实用方法。当前，有一个用于读取硬盘空余空间的方法可用。实例如下

```
package com.wj.test;

import java.io.File;
import java.io.IOException;
import java.util.List;

import org.apache.commons.io.FileUtils;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class FileUtilsTest {

    private String basePath = null;

    @Before
    public void setUp() {
        basePath = System.getProperty("user.dir") + "\\file\\";
    }

    @After
    public void tearDown() throws Exception {
    }

    /**
     * 拷贝文件
     * @throws IOException
     */
    @Test
    public void testCopy() throws IOException {
        File srcFile = new File(basePath + "a.txt");
        File destFile = new File(basePath + "b.txt");
        FileUtils.copyFile(srcFile, destFile);
    }

    /**
     * 删除文件
     * @throws IOException
     */
    @Test
    public void testDelete() throws IOException{
        File delFile = new File(basePath + "b.txt");
        FileUtils.forceDelete(delFile);
        //FileUtils.forceMkdir(delFile);
```

```
    }

    /**
     * 比较文件内容
     * @throws IOException
     */
    @Test
    public void testCompareFile() throws IOException{
        File srcFile = new File(basePath + "a.txt");
        File destFile = new File(basePath + "b.txt");
        boolean result = FileUtils.contentEquals(srcFile, destFile);
        System.out.println(result);
    }

    /**
     * 移动文件
     * @throws IOException
     */
    @Test
    public void testMoveFile() throws IOException{
        File srcFile = new File(basePath + "b.txt");
        File destDir = new File(basePath + "move");
        FileUtils.moveToDirectory(srcFile, destDir, true);
    }

    /**
     * 读取文件内容
     * @throws IOException
     */
    @Test
    public void testRead() throws IOException{
        File srcFile = new File(basePath + "a.txt");
        String content = FileUtils.readFileToString(srcFile);
        List<String> contents = FileUtils.readLines(srcFile);
        System.out.println(content);
        System.out.println("******************");
        for (String string : contents) {
            System.out.println(string);
        }
    }

    /**
     * 写入文件内容
     * @throws IOException
     */
    @Test
    public void testWrite() throws IOException{
        File srcFile = new File(basePath + "a.txt");
        FileUtils.writeStringToFile(srcFile, "\nyes文件", true);
    }

}

package com.wj.test;
```

```
import java.io.IOException;

import org.apache.commons.io.FileSystemUtils;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class FileSystemUtilsTest {
    @Before
    public void setUp() throws Exception {
    }

    @After
    public void tearDown() throws Exception {
    }

    /**
     * 获取磁盘空余空间
     * @throws IOException
     */
    @SuppressWarnings("deprecation")
    @Test
    public void testFreeSpace() throws IOException {
        // 以字节为单位
        System.out.println(FileSystemUtils.freeSpace("c:\\") / 1024 / 1024 / 1024);
        System.out.println(FileSystemUtils.freeSpace("d:\\") / 1024 / 1024 / 1024);
        // 以k为单位
        System.out.println(FileSystemUtils.freeSpaceKb("e:\\") / 1024 / 1024);
        System.out.println(FileSystemUtils.freeSpaceKb("f:\\") / 1024 / 1024);

    }

}
```

二、尾端类
不同的计算机体系结构使用不同约定的字节排序。在所谓的“低位优先”体系结构中（如Intel）低位字节处于内存中最低位置，而其后的字节，则处于更高的位置。在“高位优先”的体系结构中（Motorola），这种情况恰恰相反。
这个类库上有两个相关类：
EndianUtils包含用于交换java原对象和流之间的字节序列。
SwappedDataInputStream类是DataInput接口的一个实例。使用它，可以读取非本地的字节列。
三、行迭代器
org.apache.commons.io.LineIterator类提供了一个灵活的方式与基于行的文件交互。可以直接建一个实例，或者使用FileUtils或IOUtils的工厂方法来创建，实例如下：

```
package com.wj.test;

import java.io.File;
import java.io.IOException;

import org.apache.commons.io.FileUtils;
import org.apache.commons.io.LineIterator;
```

```
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class LineIteratorTest {

    private String basePath = null;

    @Before
    public void setUp() throws Exception {
        basePath = System.getProperty("user.dir") + "\\file\\";
    }

    @After
    public void tearDown() throws Exception {
    }

    /**
     * 测试行迭代器
     * @throws IOException
     */
    @Test
    public void testIterator() throws IOException{
        File file = new File(basePath + "a.txt");
        LineIterator li = FileUtils.lineIterator(file);
        while(li.hasNext()){
            System.out.println(li.nextLine());
        }
        LineIterator.closeQuietly(li);
    }

}
```

四、文件过滤器
org.apache.commons.io.filefilter包定义了一个合并了java.io.FileFilter以及java.io.FilenameFilt
r的接口(IOFileFilter)。除此之外，这个包还提供了一系列直接可用的IOFileFilter的实现类，可以通过
们合并其它的文件过滤器。比如，这些文件过滤器可以在列出文件时使用或者在使用文件对话框时使
。实例如下：

```
package com.wj.test;

import java.io.File;
import java.io.IOException;

import org.apache.commons.io.filefilter.EmptyFileFilter;
import org.apache.commons.io.filefilter.SuffixFileFilter;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class FileFilterTest {

    private String basePath = null;
```

```
    @Before
    public void setUp() throws Exception {
        basePath = System.getProperty("user.dir") + "\\file\\";
    }

    @After
    public void tearDown() throws Exception {
    }

    /**
     * 空内容文件过滤器
     * @throws IOException
     */
    @Test
    public void testEmptyFileFilter() throws IOException{
        File dir = new File(basePath);
        String[] files = dir.list(EmptyFileFilter.NOT_EMPTY);
        for (String file : files) {
            System.out.println(file);
        }
    }

    /**
     * 文件名称后缀过滤器
     * @throws IOException
     */
    @Test
    public void testSuffixFileFilter() throws IOException{
        File dir = new File(basePath);
        String[] files = dir.list(new SuffixFileFilter("a.txt"));
        for (String file : files) {
            System.out.println(file);
        }
    }

}

<p><strong>五、文件比较器</strong></p>
<p>org.apache.commons.io.comparator包为java.io.File提供了一些java.util.Comparator接口的现。例如，可以使用这些比较器对文件集合或数组进行排序。实例如下：</p>


package com.wj.test;

import java.io.File;
import java.io.IOException;

import org.apache.commons.io.comparator.CompositeFileComparator;
import org.apache.commons.io.comparator.DirectoryFileComparator;
import org.apache.commons.io.comparator.NameFileComparator;
import org.apache.commons.io.comparator.PathFileComparator;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
```

```
public class ComparatorTest {

    private String basePath = null;

    @Before
    public void setUp() throws Exception {
        basePath = System.getProperty("user.dir") + "\\file\\";
    }

    @After
    public void tearDown() throws Exception {
    }

    /**
     * 文件名称比较器
     * @throws IOException
     */
    @Test
    public void testNameFileComparator() throws IOException {
        File f1 = new File(basePath + "a.txt");
        File f2 = new File(basePath + "c.txt");
        int result = NameFileComparator.NAME_COMPARATOR.compare(f1, f2);
        System.out.println(result);
    }

    /**
     * 文件路径比较器
     * @throws IOException
     */
    @Test
    public void testPathFileComparator() throws IOException {
        File f1 = new File(basePath + "a.txt");
        File f2 = new File(basePath + "c.txt");
        int result = PathFileComparator.PATH_COMPARATOR.compare(f1, f2);
        System.out.println(result);
    }

    /**
     * 组合比较器
     * @throws IOException
     */
    @SuppressWarnings("unchecked")
    @Test
    public void testCompositeFileComparator() throws IOException {
        File dir = new File(basePath);
        File [] files = dir.listFiles();
        for (File file : files) {
            System.out.println(file.getName());
        }
        CompositeFileComparator cfc = new CompositeFileComparator(
                DirectoryFileComparator.DIRECTORY_COMPARATOR,
                NameFileComparator.NAME_COMPARATOR);
        cfc.sort(files);
        System.out.println("*****after sort*****");
```

```
        for (File file : files) {
            System.out.println(file.getName());
        }
    }
}
```

六、扩展流
org.apache.commons.io.input和org.apache.commons.io.output包中包含的针对数据流的各各样的的实现。包括：

空输出流 - 默默吸收发送给它的所有数据
T型输出流 - 全用两个输出流替换一个进行发送
字节数组输出流 - 这是一个更快版本的JDK类
计数流 - 计算通过的字节数
代理流 - 使用正确的方法委拖
可锁写入 - 使用上锁文件提供同步写入

代码下载地址：http://download.csdn.net/detail/tianma630/5997629

