Java 核心技术

第十一章 Java文件读写

第一节 文件系统及Java文件基本操作

华东师范大学 陈良育

文件概述

- 文件系统是由OS(操作系统)管理的
- 文件系统和Java进程是平行的，是两套系统
- 文件系统是由文件夹和文件递归组合而成
- 文件目录分隔符
  - Linux/Unix 用/隔开
  - Windows用\隔开，涉及到转义，在程序中需用/或\\代替
- 文件包括文件里面的内容和文件基本属性
- 文件基本属性：名称、大小、扩展名、修改时间等

Java 文件类File

- java.io.File是文件和目录的重要类(JDK6及以前是唯一)
  - 目录也使用File类进行表示
- File类与OS无关，但会受到OS的权限限制
- 常用方法
  - createNewFile,delete,exists, getAbsolutePath, getName, getParent,getPath, isDirectory, isFile, length, listFiles, mkdir, mkdirs
- 注意：File不涉及到具体的文件内容，只涉及属性
- 查看FileAttributeTest.java了解其用法

Java NIO

- Java 7提出的NIO包，提出新的文件系统类
  - Path, Files, DirectoryStream, FileVisitor,FileSystem
  - 是java.io.File的有益补充
    - 文件复制和移动
    - 文件相对路径
    - 递归遍历目录
    - 递归删除目录
    - ……
  - 查看相关例子

总结

- 文件系统和Java是并列的两套系统
- File类是文件基本属性操作的主要类
- Java 7提出的NIO包在某些功能上有重要的补充作用

代码(1) FileAttributeTest.java


```
import java.io.*;
public class FileAttributeTest{
  public static void main(String[] args){
    //创建目录
    File d=new File("c:/temp");
    if(!d.exists())
    {
        d.mkdirs();  //mkdir 创建单级目录 mkdirs 连续创建多级目录
    }
    System.out.println("Is d directory? " + d.isDirectory());

    //创建文件
    File f=new File("C:/temp/abc.txt");
    if(!f.exists())
    {
      try
      {
        f.createNewFile(); //创建abc.txt
      }
      catch(IOException e){ //可能会因为权限不足或磁盘已满报错
          e.printStackTrace();
      }
    }
```

代码(2) FileAttributeTest.java


```
    //输出文件相关属性
    System.out.println("Is f file? " + f.isFile());
    System.out.println("Name: "+f.getName());
    System.out.println("Parent: "+f.getParent());
    System.out.println("Path: "+f.getPath());
    System.out.println("Size: "+f.length()+" bytes");
    System.out.println("Last modified time: "+f.lastModified()+"ms");

    //遍历d目录下所有的文件信息
    System.out.println("list files in d directory");
    File[] fs = d.listFiles();  //列出d目录下所有的子文件，不包括子目录下的文件
    for(File f1:fs)
    {
        System.out.println(f1.getPath());
    }

    //f.delete(); //删除此文件
    //d.delete(); //删除目录
  }
}
```

代码(3) PathTest.java


```
import java.io.File;

public class PathTest {
    public static void main(String[] args) {
        // Path 和 java.io.File 基本类似
        // 获得path方法一,c:/temp/abc.txt
        Path path = FileSystems.getDefault().getPath("c:/temp", "abc.txt");
        System.out.println(path.getNameCount());

        // 获得path方法二，用File的toPath()方法获得Path对象
        File file = new File("c:/temp/abc.txt");
        Path pathOther = file.toPath();
        // 0,说明这两个path是相等的
        System.out.println(path.compareTo(pathOther));
```

代码(4) PathTest.java


```
        // 获得path方法三
        Path path3 = Paths.get("c:/temp", "abc.txt");
        System.out.println(path3.toString());

        // 合并两个path
        Path path4 = Paths.get("c:/temp");
        System.out.println("path4: " + path4.resolve("abc.txt"));

        if (Files.isReadable(path)) {
            System.out.println("it is readable");
        } else {
            System.out.println("it is not readable");
        }
    }
}
```

代码(5) FilesTest.java


```
import java.io.IOException;

public class FilesTest {

    public static void main(String[] a)
    {
        moveFile();
        fileAttributes();
        createDirectory();
    }

    public static void moveFile() {
        Path from = Paths.get("c:/temp", "abc.txt");
        //移动c:/temp/abc.txt到c:/temp/test/def.txt，如目标文件已存在，就替换
        Path to = from.getParent().resolve("test/def.txt");
        try {
            //文件的大小bytes
            System.out.println(Files.size(from));
            //调用文件移动方法 如果目标文件已经存在，就替换
            Files.move(from, to, StandardCopyOption.REPLACE_EXISTING);
        } catch (IOException e) {
            System.err.println("移动文件错误" + e.getMessage());
        }
    }
```

代码(6) FilesTest.java


```
public static void fileAttributes(){
    Path path = Paths.get("c:/temp");
    //1
    System.out.println(Files.isDirectory(path, LinkOption.NOFOLLOW_LINKS));
    //2
    try {
        //获得文件的基础属性
        BasicFileAttributes attributes = Files.readAttributes(path, BasicFileAttributes.class);
        System.out.println(attributes.isDirectory());
        System.out.println(new Date(attributes.lastModifiedTime().toMillis()).toLocaleString());
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

代码(7) FilesTest.java


```
public static void createDirectory(){
    Path path = Paths.get("c:/temp/test");
    try {
        //创建文件夹
        if(Files.notExists(path)){
            Files.createDirectories(path);
            System.out.println("create dir");
        }else{
            System.out.println("dir exists");
        }
        Path path2 = path.resolve("A.java");
        Path path3 = path.resolve("B.java");
        Path path4 = path.resolve("C.txt");
        Path path5 = path.resolve("D.jpg");
        Files.createFile(path2);
        Files.createFile(path3);
        Files.createFile(path4);
        Files.createFile(path5);
```

代码(8) FilesTest.java


```
            //不加条件遍历
            DirectoryStream<Path> paths = Files.newDirectoryStream(path);
            for(Path p : paths){
                System.out.println(p.getFileName());
            }
            System.out.println();

            //创建一个带有过滤器,过滤文件名以java txt结尾的文件
            DirectoryStream<Path> pathsFilter = Files.newDirectoryStream(path, "*.{java,txt}");
            for(Path p : pathsFilter){
                System.out.println(p.getFileName());
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

代码(9) SearchJPGFiles.java


```
import java.io.IOException;

class Search implements FileVisitor {

    private final PathMatcher matcher;

    public Search(String ext) {
        matcher = FileSystems.getDefault().getPathMatcher("glob:" + ext);
    }

    public void judgeFile(Path file) throws IOException {
        Path name = file.getFileName();
        if (name != null && matcher.matches(name)) {
            //文件名字已经匹配
            System.out.println("Searched file was found: " + name + " in " + file.toRealPath().toString());
        }
    }
```

代码(10) SearchJPGFiles.java


```
@Override
public FileVisitResult postVisitDirectory(Object dir, IOException exc) throws IOException {
    System.out.println("Visited: " + (Path) dir);
    return FileVisitResult.CONTINUE;
}

@Override
public FileVisitResult preVisitDirectory(Object dir, BasicFileAttributes attrs) throws IOException {
    return FileVisitResult.CONTINUE;
}

@Override
public FileVisitResult visitFile(Object file, BasicFileAttributes attrs) throws IOException {
    judgeFile((Path) file);
    return FileVisitResult.CONTINUE;
}
```

代码(11) SearchJPGFiles.java


```
    @Override
    public FileVisitResult visitFileFailed(Object file, IOException exc) throws IOException {
        // report an error if necessary
        return FileVisitResult.CONTINUE;
    }
}
//查找某一个目录下所有的jpg文件，包括子文件夹
public class SearchJPGFiles {

    public static void main(String[] args) throws IOException {
        //定义扩展名，和待查找目录
        String ext = "*.jpg";
        Path fileTree = Paths.get("C:/temp/");
        Search walk = new Search(ext);
        EnumSet<FileVisitOption> opts = EnumSet.of(FileVisitOption.FOLLOW_LINKS);

        Files.walkFileTree(fileTree, opts, Integer.MAX_VALUE, walk);

    }
}
```

谢 谢！