NIO的文件操作Path和File

2018-07-26 13:21:37 Little Programmer 阅读数 963 更多

Path

`path` 接口是在 `java7` 加入到NIO的,位于 `java.nio.file` 包,它的实例代表了文件系统里的一个路径。java中的Path表示文件系统的路径。可以指向文件或文件夹。也有相对路径和绝对路径之分。绝对路径表示从文件系统的根路径到文件或是文件夹的路径。相对路径表示从特定路径下访问指定文件或文件夹的路径。
初始化

```
Path path = Paths.get("D:\\svn-config.properties");
Path.normalize()
```

为了使用 `java.nio.file.Path` 实例，必须首先创建它。可以使用 `Paths` 类的静态方法 `Paths.get()` 来产生一个实例。
`normalize()` 方法可以标准化一个路径,就是移除掉所有在路径字符中的的 . 和 ..,同时决定路径字符串指向哪条路径.

```
Path path = Paths.get("D:\\Users\\xxx\\Desktop\\..\\svn-config.properties");
Path path1 = path.normalize();
System.out.println(path);
System.out.println(path1);
```

结果:

```
D:\Users\xxx\Desktop\..\svn-config.properties
D:\Users\xxx\svn-config.properties
```

java NIO Path类也能使用相对路径。可以通过Paths.get(basePath, relativePath)创建一个相对路径Path。示例如:

```
Path projects = Paths.get("d:\\data", "projects");
Path file = Paths.get("d:\\data", "projects\\a-project\\myfile.txt");
```

Files

Java NIO File 类提供了几个在文件系统中熟练操作文件的方法

- 判断文件存在

`LinkOption.NOFOLLOW_LINKS` 表明判断文件存在的方法不应该根据文件系统中的符号连接来判断路径是否存在

```
 Path path = Paths.get("D:\\Users\\xxx\\Desktop\\svn-config.properties");
 boolean isExist = Files.exists(path, LinkOption.NOFOLLOW_LINKS);
```

- 创建文件夹

```
Path path = Paths.get("D:\dir");
try
{
    Path newDir = Files.createDirectory(path);
} catch(FileAlreadyExistsException e){
    // the directory already exists.
} catch (IOException e) {
    //something else went wrong
    e.printStackTrace();
}
```

- 拷贝文件

```
Path sourcePath      = Paths.get("data/logging.properties");
Path destinationPath = Paths.get("data/logging-copy.properties");

try {
    Files.copy(sourcePath, destinationPath);
} catch(FileAlreadyExistsException e) {
    //destination file already exists
} catch (IOException e) {
    //something else went wrong
    e.printStackTrace();
}
```

- 覆盖文件

```
Path sourcePath      = Paths.get("data/logging.properties");
Path destinationPath = Paths.get("data/logging-copy.properties");

try {
    Files.copy(sourcePath, destinationPath,
            StandardCopyOption.REPLACE_EXISTING);
} catch(FileAlreadyExistsException e) {
    //destination file already exists
} catch (IOException e) {
    //something else went wrong
    e.printStackTrace();
}
```

- 移动文件

```
Path sourcePath      = Paths.get("data/logging-copy.properties");
Path destinationPath = Paths.get("data/subdir/logging-moved.properties");

try {
    Files.move(sourcePath, destinationPath,StandardCopyOption.REPLACE_EXISTING);
} catch (IOException e) {
    //moving file failed.
    e.printStackTrace();
}
```

- 删除文件

```
Path path = Paths.get("data/subdir/logging-moved.properties");
try {
    Files.delete(path);
} catch (IOException e) {
    //deleting file failed
    e.printStackTrace();
}
```

- 遍历文件夹

```
Path path = Paths.get("D:\\mnt");
Files.walkFileTree(path, new FileVisitor<Path>() {
    @Override
    public FileVisitResult preVisitDirectory(Path dir, BasicFileAttributes attrs) throws IOException {
        System.out.println("pre visit dir:" + dir);
        return FileVisitResult.CONTINUE;
    }

    @Override
    public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) throws IOException {
        System.out.println("visit file: " + file);
        return FileVisitResult.CONTINUE;
    }

    @Override
    public FileVisitResult visitFileFailed(Path file, IOException exc) throws IOException {
        System.out.println("visit file failed: " + file);
        return FileVisitResult.CONTINUE;
    }

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

- 查找文件

```
Path root = Paths.get("D:\\mnt");
final String fileToFind = File.separator+"path.txt";
try {
    Files.walkFileTree(root, new SimpleFileVisitor<Path>() {

        @Override
        public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) throws IOException {
            String fileString = file.toAbsolutePath().toString();
            //System.out.println("pathString = " + fileString);

            if(fileString.endsWith(fileToFind)){
                System.out.println("file found at path: " + file.toAbsolutePath());
                return FileVisitResult.TERMINATE;
            }
            return FileVisitResult.CONTINUE;
        }
    });
} catch(IOException e){
    e.printStackTrace();
}
```

- 循环删除文件

```
Path rootPath = Paths.get("data/to-delete");

try {
  Files.walkFileTree(rootPath, new SimpleFileVisitor<Path>() {
    @Override
    public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) throws IOException {
      System.out.println("delete file: " + file.toString());
      Files.delete(file);
      return FileVisitResult.CONTINUE;
    }

    @Override
    public FileVisitResult postVisitDirectory(Path dir, IOException exc) throws IOException {
      Files.delete(dir);
      System.out.println("delete dir: " + dir.toString());
      return FileVisitResult.CONTINUE;
    }
  });
} catch(IOException e){
  e.printStackTrace();
}
```

- 生成超链接

```
//连接路径
 Path p = Paths.get("D:\\mnt\\1.txt");
 //目标路径
 Path target = Paths.get("D:\\Users\\weizhiming\\Desktop");
 //生成指向目标路径的超链接,返回连接路径p
 Path p3 = Files.createSymbolicLink(p,target);
```

- 读取文件内容

```
//读取每行数据放入List中
List<String> lines = Files.readAllLines(Paths.get("d://a.txt"));
//读取文件内容放入流中
Stream<String> lines = Files.lines(Paths.get("d://a.txt"));
```