第09天 IO流

今日内容介绍

* File类
* 字符流与字节流

# File类

## File概述

打开API，搜索File类。阅读其描述：File文件和目录路径名的抽象表示形式。即，Java中把文件或者目录（文件夹）都封装成File对象。也就是说如果我们要去操作硬盘上的文件，或者文件夹只要找到File这个类即可,那么我们就要研究研究File这个类中都有那些功能可以操作文件或者文件夹呢

## File类的构造函数

File(String pathname) :将一个字符串路径封装成File对象

File(String parent,String child):传入一个父级路径和子级路径

File(File parent,String child):传入一个File类型的父级路径和子级路径

### 案例代码一:

**package** com.itheima\_01;

**import** java.io.File;

/\*

\* File:文件和目录路径名的抽象表示形式，File 类的实例是不可变的

\*

\* 构造方法：

\* File(File parent, String child)

\* File(String pathname)

\* File(String parent, String child)

\*

\*

\*/

**public** **class** FileDemo {

**public** **static** **void** main(String[] args) {

//File(String pathname) ：将指定的路径名转换成一个File对象

//File f = new File("D:\\a\\b.txt");

//File(String parent, String child) : 根据指定的父路径和文件路径创建File对象

//File f2 = new File("D:\\a","b.txt");

//File(File parent, String child) :根据指定的父路径对象和文件路径创建File对象

//File parent = new File("D:\\a");

//File f3 = new File(parent,"b.txt");

File f4 = **new** File(**new** File("D:\\a"),"b.txt");

}

}

## File类创建和删除功能

boolean createNewFile():指定路径不存在该文件时时创建文件,返回true,否则返回false

boolean mkdir():当指定的单级文件夹不存在时创建文件夹并返回true，否则返回false

boolean mkdirs():当指定的多级文件夹某一级文件夹不存在时,创建多级文件夹并返回true,否则返回false

boolean delete():删除文件或者删除单级文件夹

### 案例代码二:

**package** com.itheima\_01;

**import** java.io.File;

**import** java.io.IOException;

/\*

\* 创建功能

\* boolean createNewFile()

\* boolean mkdir()

\* boolean mkdirs()

\* 删除功能

\* boolean delete()

\*

\* 绝对路径：固定不可改变的路径，以盘符开头

\* 相对路径：相对某个参照物，不能以盘符开头

\* 在eclipse中相对路径相对应当前项目的根目录

\*

\*

\* 注意：删除一个文件夹，这个文件夹下面不能有其他的文件和文件夹

\*/

**public** **class** FileDemo2 {

**public** **static** **void** main(String[] args) **throws** IOException {

File f = **new** File("d:\\a\\b.txt");//绝对路径

File f2 = **new** File("a.txt");//相对路径

//boolean createNewFile() : 当指定文件不存在时创建文件并返回true，否则返回false

System.*out*.println(f2.createNewFile());

//boolean mkdir() : 当指定文件夹不存在时创建文件夹并返回true，否则返回false

File f3 = **new** File("b");

System.*out*.println(f3.mkdir());

//boolean mkdirs() : 创建指定文件夹，当文件夹所在的目录不存在，则顺道一块创建了

File f4 = **new** File("c\\d\\e");

System.*out*.println(f4.mkdir());

System.*out*.println(f4.mkdirs());

File f5 = **new** File("c.txt");

System.*out*.println(f5.mkdir());

//boolean delete() :当指定的文件或文件夹存在时删除文件或者文件夹 并返回true，否则返回false

System.*out*.println(f2.delete());

System.*out*.println(f3.delete());

File f6 = **new** File("c");

System.*out*.println(f6.delete());

}

}

## File类的判断功能

boolean exists():判断指定路径的文件或文件夹是否存在

boolean isAbsolute():判断当前路路径是否是绝对路径

boolean isDirectory():判断当前的目录是否存在

boolean isFile():判断当前路径是否是一个文件

boolean isHidden():判断当前路径是否是隐藏文件

### 案例代码三:

**package** com.itheima\_01;

**import** java.io.File;

**import** java.io.IOException;

/\*

\* 判断功能

\* boolean exists()

\* boolean isAbsolute()

\* boolean isDirectory()

boolean isFile()

boolean isHidden()

\*/

**public** **class** FileDemo3 {

**public** **static** **void** main(String[] args) **throws** IOException {

//method();

//method2();

//method3();

*method4*();

}

**private** **static** **void** method4() {

File f = **new** File("d:\\a\\b.txt");

//boolean isHidden() :判断File对象指向的路径是否有隐藏属性，如果隐藏了则返回true，否则返回false

System.*out*.println(f.isHidden());

}

**private** **static** **void** method3() {

File f = **new** File("a.txt");

File f2 = **new** File("b");

//boolean isDirectory() ：判断File对象指向的路径是否是文件夹，如果是则返回true，否则返回false

//boolean isFile() : 判断File对象指向的路径是否是文件，如果是则返回true，否则返回false

//System.out.println(f.isDirectory());

//System.out.println(f2.isDirectory());

System.*out*.println(f.isFile());

System.*out*.println(f2.isFile());

}

**private** **static** **void** method2() {

//File f = new File("a.txt");

File f2 = **new** File("d:\\a\b.txt");

//boolean isAbsolute() :判断File对象指向的路径是否是绝对路径，如果是绝对路径则返回true，否则返回false

System.*out*.println(f2.isAbsolute());

}

**private** **static** **void** method() {

File f = **new** File("a.txt");

//f.createNewFile();

//boolean exists() : 判断文件或者文件夹是否存在，如果存在则返回true，否则返回false

System.*out*.println(f.exists());

}

}

## File类的获取功能和修改名字功能

File getAbsoluteFile():获取文件的绝对路径,返回File对象

String getAbsolutePath():获取文件的绝对路径,返回路径的字符串

String getParent():获取当前路径的父级路径,以字符串形式返回该父级路径

File getParentFile():获取当前路径的父级路径,以字File对象形式返回该父级路径

String getName():获取文件或文件夹的名称

String getPath():获取File对象中封装的路径

long lastModified():以毫秒值返回最后修改时间

long length():返回文件的字节数

boolean renameTo(File dest): 将当前File对象所指向的路径 修改为 指定File所指向的路径

### 案例代码四:

**package** com.itheima\_01;

**import** java.io.File;

**import** java.io.IOException;

/\*

\* 获取功能

File getAbsoluteFile()

\* String getAbsolutePath()

\* String getParent()

\* File getParentFile()

\* String getName()

\* String getPath()

long lastModified()

long length()

修改功能：

boolean renameTo(File dest)

\*/

**public** **class** FileDemo4 {

**public** **static** **void** main(String[] args) **throws** IOException {

//method();

//method2();

//method3();

File f = **new** File("d.txt");

File f2 = **new** File("e.txt");

//boolean renameTo(File dest) : 将当前File对象所指向的路径 修改为 指定File所指向的路径

//注意：修改的文件路径不能存在，如果存在则修改失败

System.*out*.println(f.renameTo(f2));

}

**private** **static** **void** method3() {

File f = **new** File("a.txt");

File f2 = **new** File("d:\\a\\b.txt");

File f3 = **new** File("b");

//String getName() : 获取文件和文件夹的名称

//System.out.println(f.getName());

//System.out.println(f2.getName());

//System.out.println(f3.getName());

//String getPath() : 返回创建File对象时给的路径

//System.out.println(f.getPath());

//System.out.println(f2.getPath());

//System.out.println(f3.getPath());

//long lastModified() : 以毫秒值的形式返回最后修改时间

//System.out.println(f.lastModified());

//2514023311294

//Date d = new Date(2514023311294L);

//System.out.println(d.toLocaleString());

//long length() : 返回文件的字节数

System.*out*.println(f.length());

System.*out*.println(f3.length());

}

**private** **static** **void** method2() **throws** IOException {

//File f = new File("a.txt");

//File f2 = new File("b","c.txt");

//System.out.println(f2.createNewFile());

File parent = **new** File("b");

File f3 = **new** File(parent,"c.txt");

**if**(!parent.exists()) {

parent.mkdirs();

}

System.*out*.println(f3.createNewFile());

//String getParent()

System.*out*.println(f3.getParent());

//File getParentFile()

System.*out*.println(f3.getParentFile());

}

**private** **static** **void** method() {

File f = **new** File("d:\\a\\b.txt");

File f2 = **new** File("a.txt");

//File getAbsoluteFile() ：以File对象的形式返回当前File对象所有指向的绝对路径

System.*out*.println(f2.getAbsoluteFile());

//String getAbsolutePath() : 返回File对象所指向的绝对路径

System.*out*.println(f2.getAbsolutePath());

}

}

## File类的其它获取功能

String[] list():以字符串数组的形式返回当前路径下所有的文件和文件夹的名称

File[] listFiles():以File对象的形式返回当前路径下所有的文件和文件夹的名称

static File[] listRoots():获取计算机中所有的盘符

### 案例代码五:

**package** com.itheima\_01;

**import** java.io.File;

/\*

\* String[] list()

File[] listFiles()

static File[] listRoots()

\*/

**public** **class** FileDemo5 {

**public** **static** **void** main(String[] args) {

//method();

//method2();

//static File[] listRoots() : 返回的是所有盘符

File[] files = File.*listRoots*();

**for** (File file : files) {

System.*out*.println(file);

}

}

**private** **static** **void** method2() {

File f = **new** File("b");

File f2 = **new** File("D:\\workspace\\myFile");

File f3 = **new** File("c.txt");

//File[] listFiles()

File[] files = f3.listFiles();

**for** (File file : files) {

System.*out*.println(file.getName());

}

}

**private** **static** **void** method() {

File f = **new** File("b");

File f2 = **new** File("D:\\workspace\\myFile");

File f3 = **new** File("c.txt");

//String[] list() : 返回当前路径下所有的文件和文件夹名称

//注意：只有指向文件夹的File对象才可以调用该方法

String[] files = f3.list();

**for** (**int** i = 0; i < files.length; i++) {

System.*out*.println(files[i]);

}

}

}

## File类两个案例

### 案例代码六

输出指定目录下所有的java文件名（包含子目录）

package com.itheima\_02;

import java.io.File;

/\*

\* 需求：输出指定目录下所有的java文件名（包含子目录）

\*/

public class RecurrenceTest {

public static void main(String[] args) {

File f = new File("src");

//File f = new File("src\\com\\itheima\_01\\RecurrenceDemo.java");

method(f);

}

public static void method(File file) {

if(file.isDirectory()) {

File[] files = file.listFiles();

for (File f : files) {

//判断是否是文件对象

if(f.isFile()) {

if(f.getName().endsWith(".java")) {

System.out.println(f.getName());

}

}

else if(f.isDirectory()){

//是一个目录对象

method(f);

}

}

}

}

//输出指定目录下的所有java文件名（不包含子目录）

public static void method2(File file) {

if(file.isDirectory()) {

File[] files = file.listFiles();

for (File f : files) {

//判断是否是文件对象

if(f.isFile()) {

if(f.getName().endsWith(".java")) {

System.out.println(f.getName());

}

}

}

}

}

}

### 案例代码七

删除指定的目录（包含子目录）

package com.itheima\_02;

import java.io.File;

/\*

\* 需求：删除指定的目录（包含子目录）

\*

\* 注意：如果要删除一个目录，则需要先删除这个目录下的所有子文件和子目录

\*

\*/

public class RecurrenceTest2 {

public static void main(String[] args) {

File f = new File("d:\\a");

method(f);

}

//删除指定目录下所有文件和目录

public static void method(File file) {

if(file.isDirectory()) {

//干掉自己所有的子文件和子目录

//获取所有的子文件和子目录

File[] files = file.listFiles();

for (File f : files) {

if(f.isFile()) {

//直接干掉他

System.out.println(f.getName());

f.delete();

}

else if(f.isDirectory()) {

//继续查看是否还有文件和子目录

method(f);

}

}

//干掉自己

System.out.println(file.getName());

file.delete();

}

}

}

# 字符流与字节流

## 概述

IO流分类：

流向

输入流 读取数据 FileReader Reader

输出流 写出数据 FileWriter Writer

数据类型

字节流

字节输入流 读取数据 InputStream

字节输出流 写出数据 OutputStream

字符流

字符输入流 读取数据 Reader

字符输出流 写出数据 Writer

## 利用字符流复制文本文件

利用字符流将当前工程下的IODemo.java拷贝到d盘的a文件夹下

### 案例代码八:

**package** com.itheima\_02;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

/\*

\* 使用字符流复制文本文件

\*

\* 数据源 IODemo.java

\* 目的地 d:\\IODemo.java

\*

\*/

**public** **class** FileCopyDemo {

**public** **static** **void** main(String[] args) **throws** IOException {

//创建字符输入流对象

FileReader fr = **new** FileReader("IODemo.java");

//创建字符输出流对象

FileWriter fw = **new** FileWriter("d:\\IODemo.java");

//一次读写一个字符

/\*int ch;

while((ch = fr.read()) != -1) {

fw.write(ch);

fw.flush();

}\*/

//一次读写一个字符数组

**int** len;//用于存储读到的字符个数

**char**[] chs = **new** **char**[1024];

**while**((len = fr.read(chs)) != -1) {

fw.write(chs,0,len);

fw.flush();

}

//释放资源

fw.close();

fr.close();

}

}

## 利用字节流复制文本文件

利用字节流将当前工程下的IODemo.java拷贝到d盘下

### 案例代码九:

**package** com.itheima\_03;

**import** java.io.FileInputStream;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

/\*

\* 使用字节流复制文本文件

\*

\* 数据源 IODemo.java

\* 目的地 d:\\a\\IODemo.java

\*

\*/

**public** **class** FileCopyDemo {

**public** **static** **void** main(String[] args) **throws** IOException {

//创建字节输入流对象

FileInputStream fis = **new** FileInputStream("IODemo.java");

//创建字节输出流对象

FileOutputStream fos = **new** FileOutputStream("d:\\a\\IODemo.java");

//一次读写一个字节

/\*int by;

while((by = fis.read()) != -1) {

fos.write(by);

}\*/

//一次读写一个字节数组

**int** len;//存储读到的字节个数

**byte**[] bys = **new** **byte**[1024];

**while**((len = fis.read(bys)) != -1) {

fos.write(bys,0,len);

}

//释放资源

fos.close();

fis.close();

}

}

## 复制图片

字节流可以操作所有类型的文件,因为所有的文件在计算机中都是以字节形式存储

而字符流只能用来操作纯文本类型的文件,不能操作字节类型的文件

### 案例代码十:

**package** com.itheima\_04;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** java.io.FileOutputStream;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

/\*

\* 分别使用字符流和字节流复制图片

\*

\* 二进制文件只能使用字节流进行复制（使用windows自带记事本打开读不懂的）

\* 文本文件的复制即可使用字符流，也可以使用字节流

\*

\*/

**public** **class** CopyImage {

**public** **static** **void** main(String[] args) **throws** IOException {

//method();

//创建字节输入流对象

FileInputStream fis = **new** FileInputStream("传智学院.jpg");

//创建字节输出流对象

FileOutputStream fos = **new** FileOutputStream("d:\\传智学院.jpg");

//一次读写一个字节数组

**int** len;//用于存储读到的字节个数

**byte**[] bys = **new** **byte**[1024];

**while**((len = fis.read(bys)) != -1) {

fos.write(bys,0,len);

}

//释放资源

fos.close();

fis.close();

}

**private** **static** **void** method() **throws** FileNotFoundException, IOException {

//创建字符输入流对象

FileReader fr = **new** FileReader("传智学院.jpg");

//创建字符输出流对象

FileWriter fw = **new** FileWriter("d:\\传智学院.jpg");

//一次读写一个字符数组

**int** len;//用于存储读到的字符个数

**char**[] chs = **new** **char**[1024];

**while**((len = fr.read(chs)) != -1) {

fw.write(chs,0,len);

fw.flush();

}

//释放资源

fw.close();

fr.close();

}

}