# 1.File类

* File 类可以描述一个文件或者文件夹。

1.什么是IO流：(Input Output)？

IO流主要用于解决设备与设备之间的数据传输问题，比如 硬盘-->内存 内存-->硬盘 键盘-->内存等

1.1 IO流应用场景：

导出报表，上传照片，下载，解释xml文件...

1.2数据的保存

数据保存在内存上是临时保存，数据保存在硬盘上是永久性保存，数据一般是已文件的形式保存在硬盘上的。

1.3 sun使用了 File类来描述文件或文件夹的

## 1.路径与分割符

### 1.1 目录分割符

* 在windows上的目录分割符是 \ ,在Linux上的目录分割符是 / 。
* 注意：在windows上面，\ 与 / 都可以作为目录分割符，而且如果写/的时候,写一个即可,不需要转义符

### 1.2 路径问题

* 绝对路径：文件在硬盘上的完整路径，一般以盘符开头（上下级目录用\\表示）
* 相对路径：相对路径就是资源文件相对于当前程序所在的路径

注意：如果当前程序所在路径与资源文件路径不再同一盘符下边，是无法写相对路径的

. 当前路径

.. 上级路径

## 2、File类操作文件方法

### 2.1 File类的构造方法：

* File(String pathname)

通过将给定路径名字符串转换为抽象路径名来创建一个新 File 实例。

* File(File parent, String child)

根据 parent 抽象路径名和 child 路径名字符串创建一个新 File 实例。

* File(String parent, String child)

根据 parent 路径名字符串和 child 路径名字符串创建一个新 File 实例。

* File(URI uri)

通过将给定的 file: URI 转换为一个抽象路径名来创建一个新的 File 实例。

### 2.2 File类获取文件信息的方法

注意：以下方法与文件存在与否无关，不存在也不会报错。

* getName()：读取文件或文件夹的名称，不包含上级路径。
* getPath()：返回绝对路径，可以是相对路径，但是目录要指定。
* getAbsolutePath() ：获取文件的绝对路径
* length()：获取文件大小(字节)，如果文件不存在或者是目录，返回0
* getParent()：返回上级目录的路径名字符串，如果没有上级目录，返回null
* lastModified()：返回最后一次被修改的时间

### 2.3 File类操作文件或目录的方法

* createNewFile(path):创建新文件

在指定的位置创建一个新文件,创建成功返回true;如果文件存在返回false；如果父级目录不存在，抛出异常。

* mkdir():创建单级目录

在指定的位置创建一个新目录,只创建最底层的目录。创建成功返回true;如果目录存在返回false;如果父级目录不存在，返回false

* mkdirs():创建多级目录

在指定的位置创建一个新目录,创建多级目录。如果目录不存在，或父级目录不存在，将父级目录一同创建，成功返回true，如果目录存在，返回false。

* renameTo(File dest)：重命名或剪切目录或文件

如果目录或文件存在指定位置，进行重命名操作;如果文件或目录不存在指定位置，进行剪切操作。前提是文件或目录存在硬盘上。进行剪切操作时，不能操作非空目录。

### 2.4 File类的删除方法

* delete():删除一个文件或空文件夹。

如果文件夹不为空，则无法删除。成功返回true，失败返回false。

* deleteOnExit():

在虚拟机终止时，请求删除当前抽象路径名表示的文件或目录;保证程序异常时创建的临时文件也可以删除。

### 2.5 File类的判断方法

* exists() 文件或文件夹是否存在。
* isFile() 是否是一个文件，如果不存在，则始终为false。
* isDirectory() 是否是一个目录，如果不存在，则始终为false。
* isHidden() 是否是一个隐藏的文件或是否是隐藏的目录。
* isAbsolute() 测试此抽象路径名是否为绝对路径名。

## 3.File类操作文件夹

### 3.1 操作文件夹方法

* staic File[] listRoots()：列出所有的根目录（Window中就是所有系统的盘符）
* String[] list()：返回目录下的文件或者目录名，包含隐藏文件。对于文件这样操作会返回null。把当前文件夹下面的所有子文件名与子文件夹名 存储到一个String类型 的数组中 返回。

* listFiles()：返回目录下的文件或者目录对象（File类实例），包含隐藏文件。对于文件这样操作会返回null。把当前文件夹下面的所有子文件与子文件夹都使用了一个FIle对象描述，然后把这些File对象存储到一个FIle数组中返回。

注意：listFiles()返回的是一个File对象，可以操作文件及目录。list()返回的只是目录下所有文件及目录的名字，不可以操作文件**。建议使用listFiles()。**

* 带有过滤器的返回列表方法，过滤器可自定义。
* listFiles(FilenameFilter filter)：返回指定当前目录中符合过滤条件的子文件或子目录。对于文件这样操作会返回null。
* list(fileNameFilter filter)： 返回指定当前目录中符合过滤条件的子文件或子目录。对于文件这样操作会返回null。

### 3.2案例

**需求一**：指定一个文件夹，然后遍历该文件夹下面所有java文件

**需求二**：指定一个文件夹，然后列出文件夹下面的所有子文件与文件夹，但是格式要如下:  
 文件：  
 文件名1  
 文件名2  
 文件名3  
 ....  
 文件夹：  
 文件夹名1  
 文件夹名2  
 文件夹名3  
 ....

#### 3.2.1 使用 listFiles()

|  |
| --- |
| **public class** Dome6 {  **public static void** main(String[] args) {  File file = **new** File(**"E:\\aa\\bb"**);//指定目录位置  *listjava*(file);  System.***out***.println();  *listjava2*(file);  }   **public static void** listjava(File dir) {  File[] files = dir.listFiles();//获取文件夹下面所有子文件及文件夹  **for** (File file : files) {  String fileName = file.getName();  **if** (fileName.endsWith(**".java"**) && file.isFile()) {//判定条件：文件名以.java结尾，并且是文件  System.***out***.println(fileName);  }  }  }   **public static void** listjava2(File dir) {  File[] files = dir.listFiles();//获取文件夹下面所有子文件及文件夹  **for** (File file : files) {  String fileName = file.getName();  // 使用正则表达式匹配.java文件名  **if** (fileName.matches(**"^.+\\.java$"**) && file.isFile()) {//判定条件：文件名以.java结尾，并且是文件  System.***out***.println(fileName);  }  }  } } |

#### 3.2.2 使用listFIles(FilenameFilter filter)

|  |
| --- |
| **public class** Dome7 {  **public static void** main(String[] args) {  File file = **new** File(**"E:\\aa\\bb"**);  *listjava*(file);  }    **public static void** listjava(File dir) {  //创建File类对象，并传入过滤器  File[] files = dir.listFiles(**new** MyFilter2());  *filesIterator*(files);  }   // 遍历文件  **public static void** filesIterator(File[] files) {  System.***out***.println(**"文件："**);  **for** (File file : files) {  **if** (file.isFile()) {  System.***out***.println(file.getName());  }  }  System.***out***.println(**"文件夹："**);  **for** (File file : files) {  **if** (file.isDirectory()) {  System.***out***.println(file.getName());  }  }  }  }  // 自定义过滤器类（过滤器类需实现FilenameFilter接口） **class** MyFilter2 **implements** FilenameFilter {  // 直接在过滤器类的accept方法中返回过滤规则即可  @Override  **public boolean** accept(File dir, String name) {  //匹配文件名name以.java结尾文件  **return** name.matches(**"^.+\\.java$"**);//正则表达式  //return name.endsWith(".java");// endWith(String str)方法  } } |

# 2. IO流

## 1.File 类

* + FIle类是用来用于描述一个文件或文件夹。
  + 通过File类我们可以读取文件或文件夹的属性数据，但是如果要读取文件中的内容数据，需要IO流技术的支持。

## 2. IO流

Input(输入流) ：

Output(输出流)：

输入流与输出了流是相对的，以内存为参照物，硬盘向内存传输数据称为输入流，内存向硬盘传输数据，称为输出流。

### 2.1 IO流解决的问题

* 解决了设备与设备之间的数据传输问题。 内存🡪🡪硬盘 硬盘🡪🡪内存

### 2.2 IO流的分类

* **按照数据的流向划分**
  + 输入流
  + 输出流
* **按照处理数据的单位划分**
  + 字节流：字节流读取的数据是以字节为单位的。读取的都是文件中的二进制数据，对读取到的结果不会做任何处理（结果是ASCII编码对照表的结果）

ASCII编码对照表： 48=0 65=A 97=a

* + 字符流:字符流读取的数据是以字符为单位的。读取的也是文件中的二进制数据，不过会将这些二进制数据解码成我们能识别的字符。

字符流 = 字节流 + 解码。