Java I/O 总结

2017/03/01 | 分类： 基础技术 | 1 条评论 | 标签： io

 原文出处： linbingdong

Java中I/O操作主要是指使用Java进行输入，输出操作. Java所有的I/O机制都是基于数据流进行输入输出，这些数据流表示了字符或者字节数据的流动序列。

数据流是一串连续不断的数据的集合，就象水管里的水流，在水管的一端一点一点地供水，而在水管的另一端看到的是一股连续不断的水流。数据写入程序可以是一段、一段地向数据流管道中写入数据，这些数据段会按先后顺序形成一个长的数据流。对数据读取程序来说，看不到数据流在写入时的分段情况，每次可以读取其中的任意长度的数据，但只能先读取前面的数据后，再读取后面的数据（不能随机读取）。不管写入时是将数据分多次写入，还是作为一个整体一次写入，读取时的效果都是完全一样的。

简而言之：数据流是一组有序，有起点和终点的字节的数据序列。包括输入流和输出流。

当程序需要读取数据的时候，就会建立一个通向数据源的连接，这个数据源可以是文件，内存，或是网络连接。类似的，当程序需要写入数据的时候，就会建立一个通向目的地的连接。

数据流分类：

流序列中的数据既可以是未经加工的原始二进制数据，也可以是经一定编码处理后符合某种格式规定的特定数据。因此Java中的流分为两种： **1) 字节流：**数据流中最小的数据单元是字节 **2) 字符流：**数据流中最小的数据单元是字符， Java中的字符是Unicode编码，一个字符占用两个字节。

概览

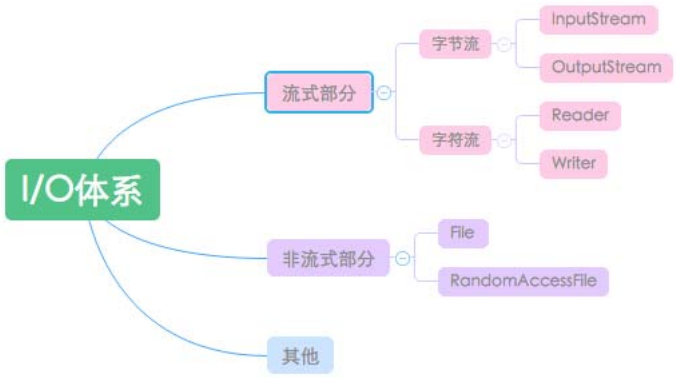
Java.io包中最重要的就是5个类和一个接口。5个类指的是File、OutputStream、InputStream、Writer、Reader；一个接口指的是Serializable。掌握了这些就掌握了Java I/O的精髓了。

Java I/O主要包括如下3层次：

1. 流式部分——最主要的部分。如：OutputStream、InputStream、Writer、Reader等
2. 非流式部分——如：File类、RandomAccessFile类和FileDescriptor等类
3. 其他——文件读取部分的与安全相关的类，如：SerializablePermission类，以及与本地操作系统相关的文件系统的类，如：FileSystem类和Win32FileSystem类和WinNTFileSystem类。

主要类如下：

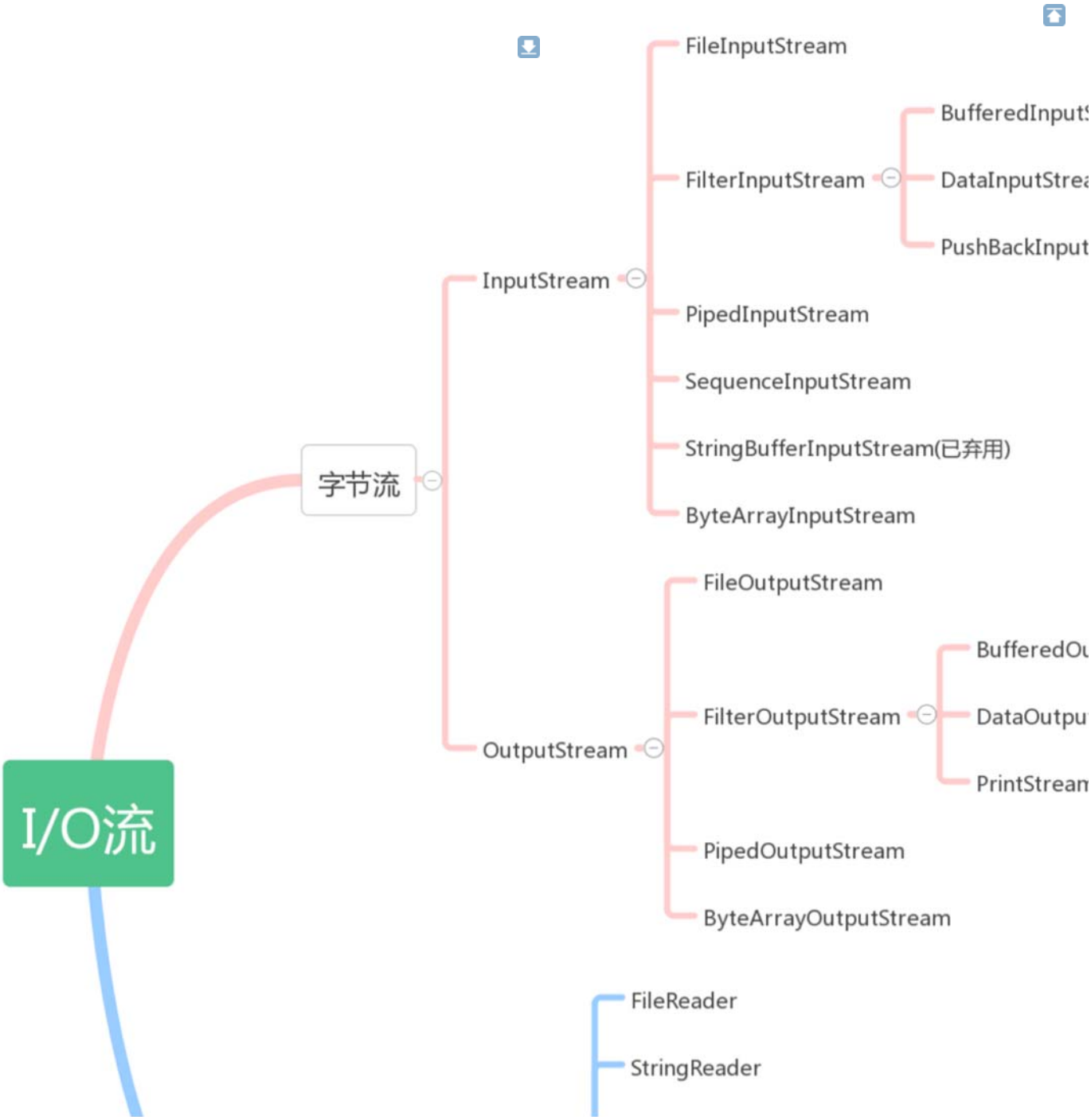
1. File（文件特征与管理）：用于文件或者目录的描述信息，例如生成新目录，修改文件名，删除文件，判断文件所在路径等。
2. InputStream（字节流，二进制格式操作）：抽象类，基于字节的输入操作，是所有输入流的父类。定义了所有输入流都具有的共同特征。
3. OutputStream（字节流，二进制格式操作）：抽象类。基于字节的输出操作。是所有输出流的父类。定义了所有输出流都具有的共同特征。
4. Reader（字符流，文本格式操作）：抽象类，基于字符的输入操作。
5. Writer（字符流，文本格式操作）：抽象类，基于字符的输出操作。
6. RandomAccessFile（随机文件操作）：它的功能丰富，**可以从文件的任意位置进行存取（输入输出）操作。**



I/O流

java.io包里有4个基本类：InputStream、OutputStream及Reader、Writer类，它们分别处理字节流和字符流。

其他各种各样的流都是由这4个派生出来的。



按来源/去向分类：

1. File（文件）： FileInputStream, FileOutputStream, FileReader, FileWriter
2. byte[]：ByteArrayInputStream, ByteArrayOutputStream
3. Char[]: CharArrayReader, CharArrayWriter
4. String: StringBufferInputStream, StringReader, StringWriter
5. 网络数据流：InputStream, OutputStream, Reader, Writer

InputStream

InputStream 为字节输入流，它本身为一个抽象类，必须依靠其子类实现各种功能，此抽象类是表示字节输入流的所有类的超类。 继承自 InputStream 的流都是向程序中输入数据的，且数据单位为字节（8bit）；

InputStream是输入字节数据用的类，所以InputStream类提供了3种重载的read方法.Inputstream类中的常用方法：

- public abstract int read()：读取一个byte的数据，返回值是高位补0的int类型值。若返回值=-1说明没有读取到任何字节读取工作结束。
- public int read(byte b[ ])：读取b.length个字节的数据放到b数组中。返回值是读取的字节数。该方法实际上是调用下一个方法实现的
- public int read(byte b[ ], int off, int len)：从输入流中最多读取len个字节的数据，存放到偏移量为off的b数组中。
- public int available()：返回输入流中可以读取的字节数。注意：若输入阻塞，当前线程将被挂起，如果InputStream对象调用这个方法的话，它只会返回0，这个方法必须由继承InputStream类的子类对象调用才有用，
- public long skip(long n)：忽略输入流中的n个字节，返回值是实际忽略的字节数, 跳过一些字节来读取
- public int close( ) ：使用完后，必须对我们打开的流进行关闭。

来看看几种不同的InputStream：

1. FileInputStream把一个文件作为InputStream，实现对文件的读取操作
2. ByteArrayInputStream：把内存中的一个缓冲区作为InputStream使用
3. StringBufferInputStream：把一个String对象作为InputStream
4. PipedInputStream：实现了pipe的概念，主要在线程中使用
5. SequenceInputStream：把多个InputStream合并为一个InputStream

OutputStream

OutputStream提供了3个write方法来做数据的输出，这个是和InputStream是相对应的。

- public void write(byte b[ ])：将参数b中的字节写到输出流。
- public void write(byte b[ ], int off, int len) ：将参数b的从偏移量off开始的len个字节写到输出流。
- public abstract void write(int b) ：先将int转换为byte类型，把低字节写入到输出流中。
- public void flush( ) : 将数据缓冲区中数据全部输出，并清空缓冲区。
- public void close( ) : 关闭输出流并释放与流相关的系统资源。

几种不同的OutputStream：

1. ByteArrayOutputStream：把信息存入内存中的一个缓冲区中
2. FileOutputStream：把信息存入文件中
3. PipedOutputStream：实现了pipe的概念，主要在线程中使用
4. SequenceOutputStream：把多个OutStream合并为一个OutStream

Reader和InputStream类似；Writer和OutputStream类似。

有两个需要注意的：

1. **InputStreamReader** ： 从输入流读取字节，在将它们转换成字符。
2. **BufferReader :**接受Reader对象作为参数，并对其添加字符缓冲器，使用readline()方法可以读取一行。

如何选择I/O流

1. 确定是输入还是输出
   输入:输入流 InputStream Reader
   输出:输出流 OutputStream Writer
2. 明确操作的数据对象是否是纯文本
   是:字符流 Reader，Writer
   否:字节流 InputStream，OutputStream
3. 明确具体的设备。
   - 文件：
     读：FileInputStream,, FileReader,
     写：FileOutputStream，FileWriter
   - 数组：
     byte[ ]：ByteArrayInputStream, ByteArrayOutputStream
     char[ ]：CharArrayReader, CharArrayWriter
   - String：
     StringBufferInputStream(已过时，因为其只能用于String的每个字符都是8位的字符串), StringReader, StringWriter
   - Socket流
     键盘：用System.in（是一个InputStream对象）读取，用System.out（是一个OutoutStream对象）打印
4. 是否需要转换流
   是，就使用转换流，从Stream转化为Reader、Writer：InputStreamReader，OutputStreamWriter
5. 是否需要缓冲提高效率
   是就加上Buffered：BufferedInputStream, BufferedOuputStream, BufferedReader, BufferedWriter
6. 是否需要格式化输出

示例代码

- 将标准输入（键盘输入）显示到标准输出（显示器），支持字符。

```
char ch;
BufferedReader in = new BufferedReader(new InputStreamReader(System.in));  //将字节流转为字符流，带缓冲
try {
    while ((ch = (char) in.read()) != -1){
        System.out.print(ch);
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

- 将AtomicityTest.java的内容打印到显示器

方法一：

```
BufferedReader in = new BufferedReader(new FileReader("AtomicityTest.java"));
String s;
try {
    while ((s = in.readLine()) != null){
        System.out.println(s);
    }
    in.close();
} catch (IOException e) {
    e.printStackTrace();
}
```

方法二：

```
FileReader in = new FileReader("AtomicityTest.java");
int b;
try {
    while ((b = in.read()) != -1){
        System.out.print((char)b);
    }
    in.close();
} catch (IOException e) {
    e.printStackTrace();
}
```

方法三：(有可能出现乱码）

```
FileInputStream in = new FileInputStream("AtomicityTest.java");
int n = 50;
byte[] buffer = new byte[n];
try {
    while ((in.read(buffer,0,n) != -1 && n > 0)){
        System.out.print(new String(buffer));
    }
    in.close();
} catch (IOException e) {
    e.printStackTrace();
}
```

- 将文件A的内容拷贝到文件B

```
FileInputStream in = new FileInputStream("AtomicityTest.java");
FileOutputStream out = new FileOutputStream("copy.txt");
int b;
while ((b = in.read()) != -1){
    out.write(b);
}
out.flush();
in.close();
```

```
9 out.close();
```

- 将标准输入的内容写入文件

```
1 Scanner in = new Scanner(System.in);
2 FileWriter out = new FileWriter("systemIn.log");
3 String s;
4 while (!(s = in.nextLine()).equals("Q")){
5     out.write(s + "\n");
6 }
7 out.flush();
8 out.close();
9 in.close();
```

21

相关文章

- Java 标准 I/O 流编程一览笔录
- 理解Java中字符流与字节流的区别
- 也谈IO模型
- Java 编程要点之 I/O 流详解
- Java I/O 模型的演进
- 【Java TCP/IP Socket】Java NIO Socket VS 标准IO Socket
- java中的IO整理
- Java NIO系列教程（12）：Java NIO与IO
- java中的IO整理
- Java I/O 操作及优化建议

发表评论

Comment form

Name*

姓名

邮箱*

请填写邮箱

网站 (请以 http://开头)

请填写网站地址

评论内容*

请填写评论内容

(*) 表示必填项

提交评论

1 条评论

1. *wang* 说道：
   2017/03/08 下午 1:53

   不错

   0 0

   回复

« 保障服务的持续高可用、高性能及负载均衡
关于Java Collections的几个常见问题 »

- 本周热门文章
- 本月热门
- 热门标签

0 记一次集群内无可用 http 服务问题...

1 Java 技术之垃圾回收机制

2 公司编程竞赛之最长路径问题

3 Java 中的十个"单行代码编程"（O...

4 Java 中 9 个处理 Exception ...

5 HttpClient 以及 Json 传递的...

6 浅析 Spring 中的事件驱动机制

7 浅析分布式下的事件驱动机制（ PubS...

8 探索各种随机函数（Java 环境...

9 Java 守护线程概述

最新评论

- Re: 攻破JAVA NIO技术壁垒
  Hi，请到伯乐在线的小组发帖提问，支持微信登录。链接是： http://group.jobbole.... 唐尤华
- Re: 攻破JAVA NIO技术壁垒
  TCP服务端的NIO写法 服务端怎么发送呢。原谅小白 菜鸟
- Re: 关于 Java 中的 double check ...
  volatile 可以避免指令重排啊。所以double check还是可以用的。 hipilee
- Re: Spring4 + Spring MVC + M...
  Hi，请到伯乐在线的小组发帖提问，支持微信登录。链接是： http://group.jobbole.... 唐尤华
- Re: Spring4 + Spring MVC + M...
  我的一直不太明白，spring的bean容器和springmvc的bean容器之间的关系。 hw_绝影
- Re: Spirng+SpringMVC+Maven+Myba...
  很好，按照步骤，已经成功。 莫凡
- Re: Spring中@Transactional事务...
  声明式事务可以用aop来实现,分别是jdk代理和cglib代理,基于接口和普通类.在同一个类中一个方... chengjiliang
- Re: 关于 Java 中的 double check ...
  在JDK1.5之后，用volatile关键字修饰_INSTANCE属性 就能避免因指令重排导致的对象... Byron

关于ImportNew

ImportNew 专注于 Java 技术分享。于2012年11月11日 11:11正式上线。是的，这是一个很特别的时刻 :)

ImportNew 由两个 Java 关键字 import 和 new 组成，意指：Java 开发者学习新知识的网站。 import 可认为是学习和吸收， new 则可认为是新知识、新技术圈子和新朋友……

-

联系我们

Email：ImportNew.com@gmail.com
新浪微博：@ImportNew
推荐微信号

反馈建议：ImportNew.com@gmail.com
广告与商务合作QQ：2302462408

推荐关注

小组 – 好的话题、有启发的回复、值得信赖的圈子
头条 – 写了文章？看干货？去头条！
相亲 – 为IT单身男女服务的征婚传播平台
资源 – 优秀的工具资源导航
翻译 – 活跃 & 专业的翻译小组
博客 – 国内外的精选博客文章
设计 – UI,网页，交互和用户体验
前端 – JavaScript, HTML5, CSS
安卓 – 专注Android技术分享
iOS – 专注iOS技术分享
Java – 专注Java技术分享
Python – 专注Python技术分享