搜索

找找看

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

随笔分类(150)

Android
C/C++(18)
Hibernate(16)
JAVA(59)
Java EE(6)
Oracle(1)
PHP(2)
Shell(1)
software(1)
Spring(14)
Spring Boot(3)
高性能高并发(5)
工具使用(2)
考研英语(2)
面试笔记(12)
设计模式
数据库(3)
随意写写(1)
系统架构(3)
项目管理
异常问题(1)

随笔档案(133)

2017年7月 (1)
2017年6月 (2)
2017年5月 (4)
2017年4月 (3)
2017年3月 (9)

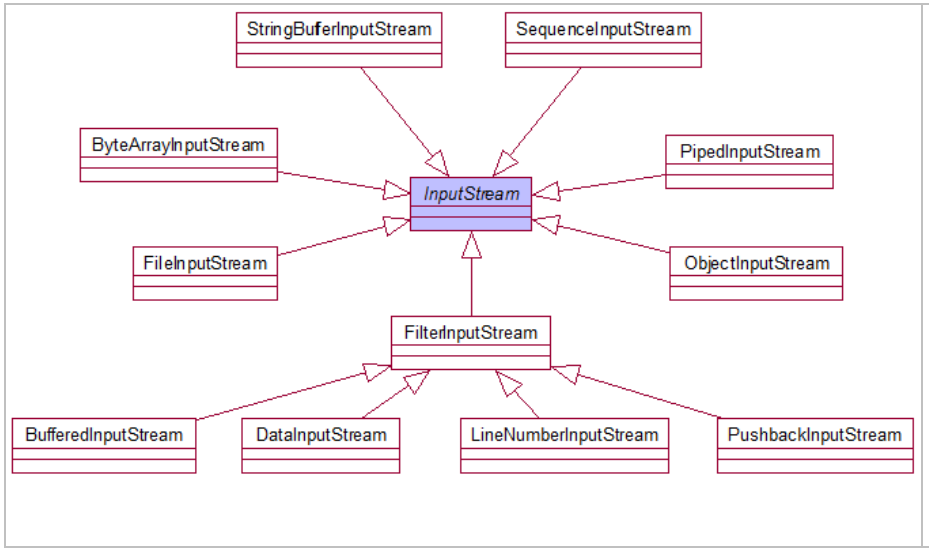
JAVA基础知识之IO——Java IO体系及常用类

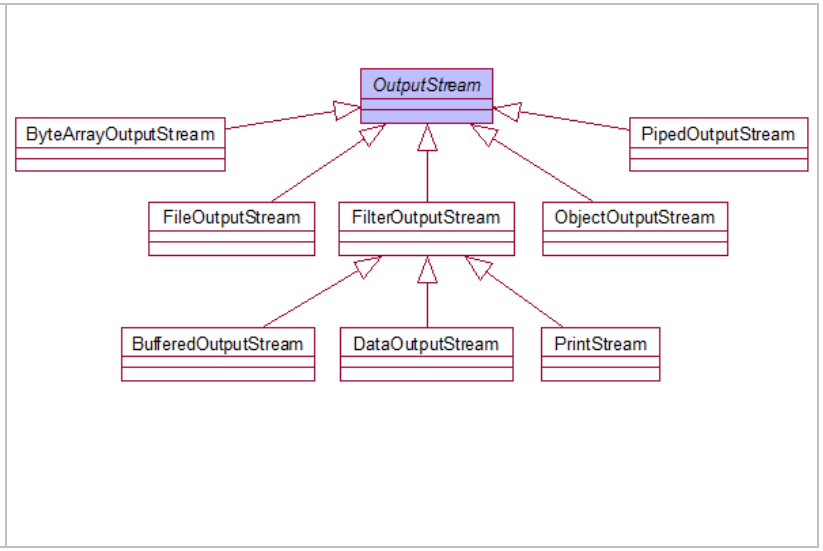
Java IO体系

个人觉得可以用"字节流操作类和字符流操作类组成了Java IO体系"来高度概括Java IO体系。

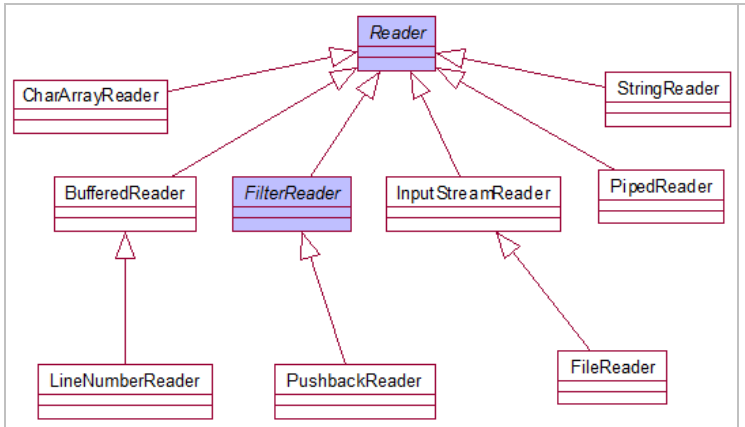
借用几张网络图片来说明(图片来自 http://blog.csdn.net/zhangerqing/article/details/8466532)

- **基于字节的IO操作**





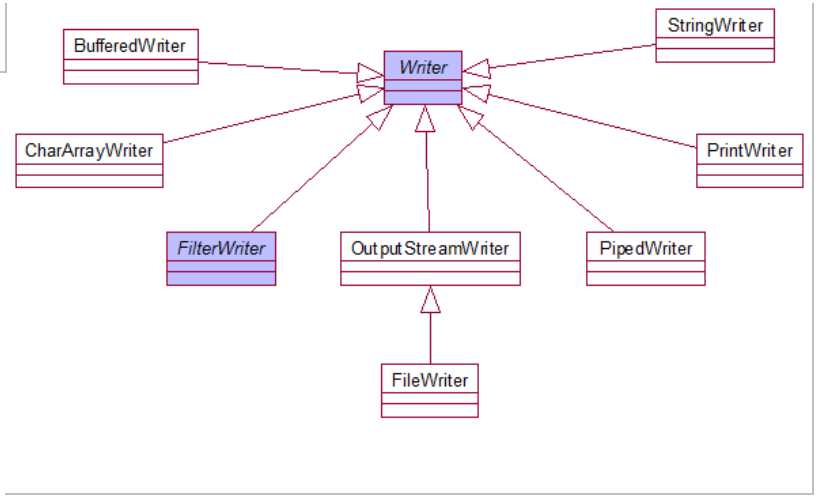
- **基于字符的IO操作**



2017年2月 (26)
2017年1月 (11)
2016年12月 (19)
2016年11月 (28)
2016年10月 (8)
2016年9月 (3)
2015年10月 (5)
2015年9月 (13)
2015年8月 (1)

学习资源

EJB到底是什么，真的那么神秘吗？？
SSH框架网上商城项目第1战之整合Struts2、Hibernate4.3和Spring4.2
Tag列表
极客学院-JavaWeb开发知识体系图
极客学院-项目实战



从上图可以看到，整个Java IO体系都是基于字符流(**InputStream/OutputStream**) 和 字节流(**Reader/Writer**)作为基类，根据不同的数据载体或功能派生出来的。

IO常用类

- **文件流：FileInputStream/FileOutputStream， FileReader/FileWriter**

这四个类是专门操作文件流的，用法高度相似，区别在于前面两个是操作字节流，后面两个是操作字符流。它们都会直接操作文件流，直接与OS底层交互。因此他们也被称为**节点流。**

注意使用这几个流的对象之后，需要关闭流对象，因为java垃圾回收器不会主动回收。不过在Java7之后，可以在 try() 括号中打开流，最后程序会自动关闭流对象，不再需要显示地close。

下面演示这四个流对象的基本用法，

```
1 package io;
2
3 import java.io.FileInputStream;
4 import java.io.FileNotFoundException;
5 import java.io.FileOutputStream;
6 import java.io.FileReader;
7 import java.io.FileWriter;
8 import java.io.IOException;
9
10 public class TestIO {
11     public static void FileInputStreamTest() throws IOException {
12         FileInputStream fis = new FileInputStream("tmp2.txt");
13         byte[] buf = new byte[1024];
14         int hasRead = 0;
15
16         //read()返回的是单个字节数据（字节数据可以直接专程int类型），但是read(buf)返回的是读取到的字节数，真正的数据保存在buf中
17         while ((hasRead = fis.read(buf)) > 0) {
18             //每次最多将1024个字节转换成字符串，这里tmp2.txt中的字符小于1024，所以一次就读完了
19             //循环次数 = 文件字符数 除以 buf长度
```

```
20            System.out.println(new String(buf, 0 ,hasRead));
21            /*
22             * 将字节强制转换成字符后逐个输出，能实现和上面一样的效果。但是如果源文件是中文的话可能会乱码
23
24            for (byte b : buf)    {
25                char ch = (char)b;
26                if (ch != '\r')
27                System.out.print(ch);
28            }
29            */
30        }
31        //在finally块里close更安全
32        fis.close();
33    }
34
35    public static void FileReaderTest() throws IOException {
36
37        try (
38                // 在try() 中打开的文件， JVM会自动关闭
39                FileReader fr = new FileReader("tmp2.txt")) {
40            char[] buf = new char[32];
41            int hasRead = 0;
42            // 每个char都占两个字节，每个字符或者汉字都是占2个字节，因此无论buf长度为多少，总是能读取中文字符长度的整数倍，不会乱码
43            while ((hasRead = fr.read(buf)) > 0) {
44                // 如果buf的长度大于文件每行的长度，就可以完整输出每行，否则会断行。
45                // 循环次数 = 文件字符数 除以 buf长度
46                System.out.println(new String(buf, 0, hasRead));
47                // 跟上面效果一样
48                // System.out.println(buf);
49            }
50        } catch (IOException ex) {
51            ex.printStackTrace();
52        }
53    }
54
55    public static void FileOutputStreamTest() throws FileNotFoundException, IOException {
56        try (
57                //在try()中打开文件会在结尾自动关闭
58                FileInputStream fis = new FileInputStream("tmp2.txt");
59                FileOutputStream fos = new FileOutputStream("tmp3.txt");
60                ) {
61            byte[] buf = new byte[4];
62            int hasRead = 0;
63            while ((hasRead = fis.read(buf)) > 0) {
64                //每读取一次就写一次，读多少就写多少
65                fos.write(buf, 0, hasRead);
66            }
67            System.out.println("write success");
68        } catch (IOException e) {
69            e.printStackTrace();
```

```
70          }
71      }
72
73      public static void FileWriterTest() throws IOException {
74          try (FileWriter fw = new FileWriter("tmp4.txt")) {
75              fw.write("天王盖地虎\r\n");
76              fw.write("宝塔镇河妖\r\n");
77          } catch (IOException e) {
78              e.printStackTrace();
79          }
80      }
81      public static void main(String[] args) throws IOException {
82          //FileInputStreamTest();
83          //FileReaderTest();
84          //FileOutputStreamTest();
85          FileWriterTest();
86      }
87  }
```

- **包装流：PrintStream/PrintWriter/Scanner**

PrintStream可以封装（包装）直接与文件交互的节点流对象OutputStream，使得编程人员可以忽略设备底层的差异，进行一致的IO操作。因此这种流也称为处理流或者包装流。

PrintWriter除了可以包装字节流OutputStream之外，还能包装字符流Writer

Scanner可以包装键盘输入，方便地将键盘输入的内容转换成我们想要的数据类型。

- **字符串流：StringReader/StringWriter**

这两个操作的是专门操作String字符串的流，其中StringReader能从String中方便地读取数据并保存到char数组，而StringWriter则将字符串类型的数据写入到StringBuffer中（因为String不可写）。

- **转换流：InputStreamReader/OutputStreamReader**

这两个类可以将字节流转换成字符流，被称为字节流与字符流之间的桥梁。我们经常在读取键盘输入(System.in)或网络通信的时候，需要使用这两个类

- **缓冲流：BufferedReader/BufferedWriter ， BufferedInputStream/BufferedOutputStream**

Oracle官方的描述：

Most of the examples we've seen so far use unbuffered I/O. This means each read or write request is handled directly by the underlying OS. This can make a program much less efficient.

Buffered input streams read data from a memory area known as a buffer; the native input API is called only when the buffer is empty. Similarly, buffered output streams write data to a buffer, and the native output API is called only when the buffer is full.

即，

没有经过Buffered处理的IO， 意味着每一次读和写的请求都会由OS底层直接处理，这会导致非常低效的问题。

经过Buffered处理过的输入流将会从一个buffer内存区域读取数据，本地API只会在buffer空了之后才会被调用（可能一次调用会填充很多数据进buffer）。

经过Buffered处理过的输出流将会把数据写入到buffer中，本地API只会在buffer满了之后才会被调用。

BufferedReader/BufferedWriter可以将字符流(Reader)包装成缓冲流，这是最常见用的做法。

另外，**BufferedReader提供一个readLine()可以方便地读取一行**，而FileInputStream和FileReader只能读取一个字节或者一个字符，

因此BufferedReader也被称为行读取器

下面演示上面提到的常见类，

```
1 package io;
2
3 import java.io.BufferedReader;
4 import java.io.FileInputStream;
5 import java.io.FileNotFoundException;
6 import java.io.FileOutputStream;
7 import java.io.FileReader;
8 import java.io.IOException;
9 import java.io.InputStreamReader;
10 import java.io.PrintStream;
11 import java.io.PushbackReader;
12 import java.io.StringReader;
13 import java.io.StringWriter;
14
15 public class TestIO {
16     public static void printStream() throws FileNotFoundException, IOException {
17         try (
18                 FileOutputStream fos = new FileOutputStream("tmp.txt");
19                 PrintStream ps = new PrintStream(fos)) {
20             ps.println("普通字符串\n");
21             //输出对象
22             ps.println(new TestIO());
23         } catch (IOException e) {
24             e.printStackTrace();
25         }
26         System.out.println("输出完成");
27
28     }
29     public static void stringNode() throws IOException {
30         String str = "天王盖地虎\n"
31                 + "宝塔镇河妖\n";
32         char[] buf = new char[32];
33         int hasRead = 0;
34         //StringReader将以String字符串为节点读取数据
35         try (StringReader sr = new StringReader(str)) {
36             while ((hasRead = sr.read(buf)) > 0) {
37                 System.out.print(new String(buf, 0, hasRead));
38             }
39         } catch (IOException e) {
40             e.printStackTrace();
```

```
41          }
42
43          //由于String是一个不可变类，因此创建StringWriter时，实际上是以一个StringBuffer作为输出节点
44          try (StringWriter sw = new StringWriter()) {
45              sw.write("黑夜给了我黑色的眼睛\n");
46              sw.write("我却用它寻找光明\n");
47              //toString()返回sw节点内的数据
48              System.out.println(sw.toString());
49          } catch (IOException e) {
50              e.printStackTrace();
51          }
52      }
53
54      public static void keyIn() throws IOException {
55          try (
56                  //InputStreamReader是从byte转成char的桥梁
57                  InputStreamReader reader = new InputStreamReader(System.in);
58                  //BufferedReader(Reader in)是char类型输入的包装类
59                  BufferedReader br = new BufferedReader(reader);
60                  ) {
61              String line = null;
62              while ((line = br.readLine()) != null) {
63                  if (line.equals("exit")) {
64                      //System.exit(1);
65                      break;
66                  }
67                  System.out.println(line);
68              }
69          } catch (IOException e) {
70              e.printStackTrace();
71          }
72      }
73
74      public static void pushback() throws FileNotFoundException, IOException {
75          try (PushbackReader pr = new PushbackReader(new FileReader("C:/PROJECT/JavaBasic/PROJECT_JavaBasic/src/io/TestIO.java"),64)) {
76              char[] buf = new char[32];
77              String lastContent = "";
78              int hasRead = 0;
79              while ((hasRead = pr.read(buf)) > 0) {
80                  String content = new String(buf, 0, hasRead);
81                  int targetIndex = 0;
82                  if ((targetIndex = (lastContent + content).indexOf("targetIndex = (lastContent + content)")) > 0) {
83                      pr.unread((lastContent + content).toCharArray());
84                      if (targetIndex > 32) {
85                          buf = new char[targetIndex];
86                      }
87                      pr.read(buf , 0 , targetIndex);
88                      System.out.println(new String(buf, 0 , targetIndex));
89                      System.exit(0);
90                  } else {
```

```
91                      System.out.println(lastContent);
92                      lastContent = content;
93                  }
94              }
95          } catch (IOException e) {
96              e.printStackTrace();
97          }
98      }
99
100     public static void main(String[] args) throws IOException {
101         printStream();
102         //stringNode();
103         //keyIn();
104         //pushback();
105     }
106 }
```

总结上面几种流的应用场景：

- FileInputStream/FileOutputStream 需要逐个字节处理原始二进制流的时候使用，效率低下
- FileReader/FileWriter 需要组个字符处理的时候使用
- StringReader/StringWriter 需要处理字符串的时候，可以将字符串保存为字符数组
- PrintStream/PrintWriter 用来包装FileOutputStream 对象，方便直接将String字符串写入文件
- Scanner 用来包装System.in流，很方便地将输入的String字符串转换成需要的数据类型
- InputStreamReader/OutputStreamReader， 字节和字符的转换桥梁，在网络通信或者处理键盘输入的时候用
- BufferedReader/BufferedWriter ， BufferedInputStream/BufferedOutputStream ， 缓冲流用来包装字节流后者字符流，提升IO性能，BufferedReader还可以方便地读取一行，简化编程。

分类: JAVA

fysola
关注 - 3
粉丝 - 7
+加关注

« 上一篇：JAVA基础知识之IO——IO流(Stream)的概念
» 下一篇：JAVA基础知识之IO——对象序列化

posted @ 2016-12-01 22:53 fysola 阅读(5554) 评论(0) 编辑 收藏