I/O 框架

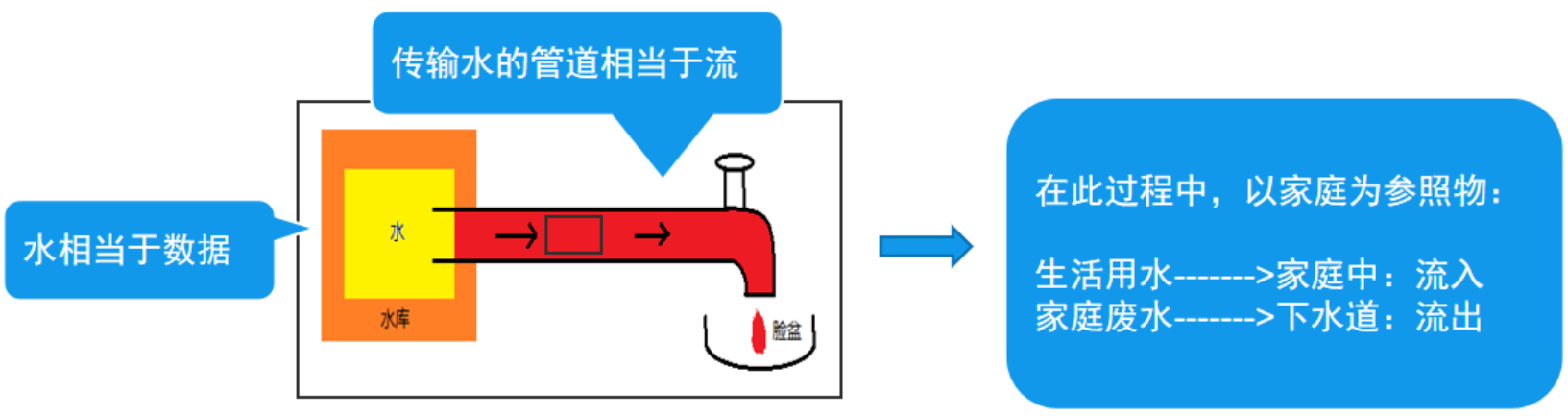
> Author:zhangzhang
>
> Version：1.0.0

一、流

1.1 概念

> - 内存与存储设备之间传输数据的通道。
> - 水借助管道传输；数据借助流传输。

生活中流



1.2 流的分类

1.2.1 按方向【 重点 】

> - 输入流：将<存储设备>中的内容读入到<内存>中。
> - 输出流：将<内存>中的内容写入到<存储设备>中。

1.2.2 按单位

> - 字节流：以字节为单位，可以读写所有数据 。
> - 字符流：以字符为单位，只能读写文本数据 。

1.2.3 按功能

> - 节点流：具有实际传输数据的读写功能。
> - 过滤流：在节点流的基础之上增强功能。

二、字节流【 重点 】

2.1 字节抽象类

> InputStream：字节输入流
>
> - public int read(){}。
> - .public int read(byte[] b){}。
> - public int read(byte[] b,int off,int len){}。

> OutputStream：字节输出流
>
> - public void write(int n){}。
> - public void write(byte[] b){}。
> - public void write(byte[] b,int off,int len){}。

2.2 字节节点流

> FileOutputStream：
>
> - public void write(byte[] b)。
> - 一次写多个字节，将b数组中所有字节，写入输出流。

> FileInputStream：
>
> - public int read(byte[] b)。
> - 从流中读取多个字节，将读到内容存入b数组，返回实际读到的字节数。
> - 如果达到文件的尾部，则返回-1。

> 案例演示：FileInputStream读取文件。

```
public class TestFileInputStream {
  public static void main(String[] args) throws Exception{
    //1创建FileInputStream,并指定文件路径
    FileInputStream fis=new FileInputStream("d:\\aaa.txt");
    //2读取文件
    //fis.read()
    //2.1单个字节读取
    //int data=0;
    //while((data=fis.read())!=-1) {
    //  System.out.print((char)data);
    //}
    //2.2一次读取多个字节
    byte[] buf=new byte[1024];
    int count=0;
    while((count=fis.read(buf))!=-1) {
      System.out.println(new String(buf,0,count));
    }

    //3关闭
    fis.close();
    System.out.println();
    System.out.println("执行完毕");
  }
}
```

> 案例演示：FileOutputStream写入文件。

```
public class TestFileOutputStream {
  public static void main(String[] args) throws Exception{
    //1创建文件字节输出流对象
    FileOutputStream fos=new FileOutputStream("d:\\bbb.txt",true);
    //2写入文件
    //fos.write(97);
    //fos.write('b');
    //fos.write('c');
    String string="helloworld";
    fos.write(string.getBytes());
    //3关闭
    fos.close();
    System.out.println("执行完毕");
  }
}
```

> 案例演示：使用字节流复制文件。

```
public class TestCopy {
  public static void main(String[] args) throws Exception{
    //1创建流
    //1.1文件字节输入流
    FileInputStream fis=new FileInputStream("d:\\001.jpg");
    //1.2文件字节输出流
    FileOutputStream fos=new FileOutputStream("d:\\002.jpg");
    //2一边读，一边写
    byte[] buf=new byte[1024];
```

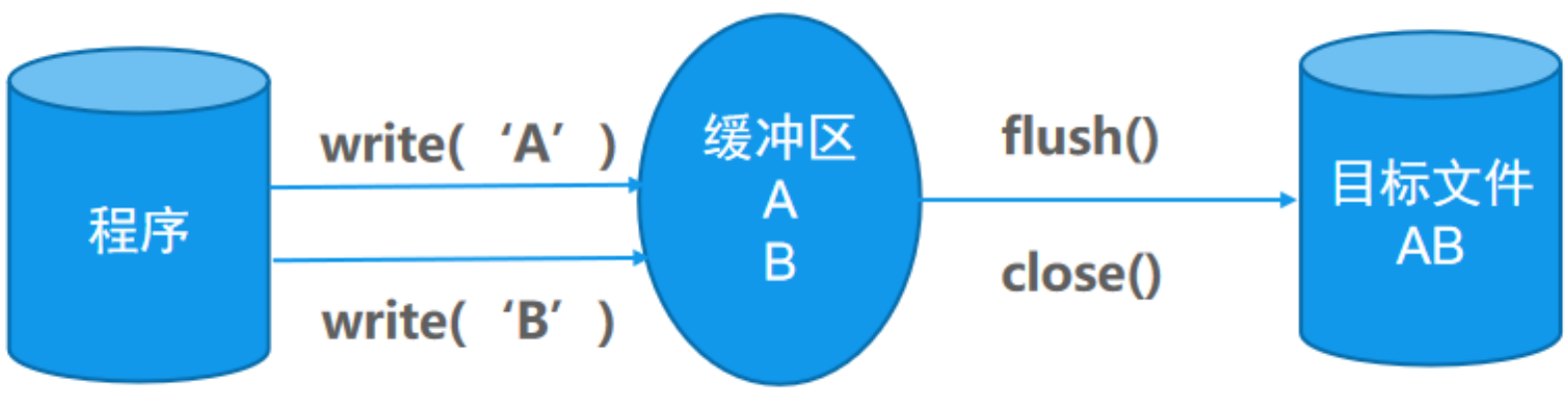
```
    int count=0;
    while((count=fis.read(buf))!=-1) {
      fos.write(buf,0,count);
    }
    //3关闭
    fis.close();
    fos.close();
    System.out.println("复制完毕");


  }
}
```

2.3 字节缓冲流

> 缓冲流：BufferedOutputStream/BufferedInputStream
>
> - 提高IO效率，减少访问磁盘的次数。
> - 数据存储在缓冲区中，flush是将缓存区的内容写入文件中，也可以直接close。

字节缓冲流



> 案例演示：

```
public class TestBufferedInputStream {
  public static void main(String[] args) throws Exception{
    //1创建BufferedInputStream
    FileInputStream fis=new FileInputStream("d:\\aaa.txt");
    BufferedInputStream bis=new BufferedInputStream(fis);
    //2读取
        //2.1单个字节读取
    //int data=0;
    //while((data=bis.read())!=-1) {
    //  System.out.print((char)data);
    //}
    //2.2一次读取多个字节
    byte[] buf=new byte[1024];
    int count=0;
    while((count=bis.read(buf))!=-1) {
      System.out.println(new String(buf,0,count));
    }

    //3关闭
    bis.close();java

  }
}
```

```
public class TestBufferedOutputStream {
  public static void main(String[] args) throws Exception{
    //1创建字节输出缓冲流
    FileOutputStream fos=new FileOutputStream("d:\\buffer.txt");
    BufferedOutputStream bos=new BufferedOutputStream(fos);
    //2写入文件
    for(int i=0;i<10;i++) {
      bos.write("helloworld\r\n".getBytes());//写入8K缓冲区
      bos.flush();//刷新到硬盘
    }
    //3关闭(内部调用flush方法)
    bos.close();


  }
}
```

2.4 对象流

> 对象流：ObjectOutputStream/ObjectInputStream
>
> - 增强了缓冲区功能。

- 增强了读写8种基本数据类型和字符串功能。
- 增强了读写对象的功能：
  - readObject() 从流中读取一个对象。
  - writeObject(Object obj) 向流中写入一个对象。

注：使用流传输对象的过程称为序列化、反序列化。

> 案例演示：使用对象流实现序列化和反序列化。

```
public class Student implements Serializable{

  /**
   * serialVersionUID:序列化版本号ID,
   */
  private static final long serialVersionUID = 100L;
  private String name;
  private transient int age;

  public static String country="中国";

  public Student() {
    // TODO Auto-generated constructor stub
  }
  public Student(String name, int age) {
    super();
    this.name = name;
    this.age = age;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public int getAge() {
    return age;
  }
  public void setAge(int age) {
    this.age = age;
  }
  @Override
  public String toString() {
    return "Student [name=" + name + ", age=" + age + "]";
  }
}
```

```
public class TestSerializable {
  public static void main(String[] args) throws Exception{
    //1创建对象流
    FileOutputStream fos=new FileOutputStream("d:\\stu.bin");
    ObjectOutputStream oos=new ObjectOutputStream(fos);
    //2序列化(写入操作)
    Student zhangsan=new Student("张三", 20);
    Student lisi=new Student("李四", 22);
    ArrayList<Student> list=new ArrayList<>();
    list.add(zhangsan);
    list.add(lisi);
    oos.writeObject(list);

    //3关闭
    oos.close();
    System.out.println("序列化完毕");
  }
}
```

```
public class TestDeSerializable  {
  public static void main(String[] args) throws Exception {
    //1创建对象流
    FileInputStream fis=new FileInputStream("d:\\stu.bin");
    ObjectInputStream ois=new ObjectInputStream(fis);
    //2读取文件(反序列化)
    //Student s=(Student)ois.readObject();
    //Student s2=(Student)ois.readObject();
    ArrayList<Student> list=(ArrayList<Student>)ois.readObject();
    //3关闭
    ois.close();
    System.out.println("执行完毕");
    //System.out.println(s.toString());
    //System.out.println(s2.toString());
    System.out.println(list.toString());
  }
}
```

> 对象序列化的细节：

- 必须实现Serializable接口。
- 必须保证其所有属性均可序列化。
- transient修饰为临时属性，不参与序列化。
- 读取到文件尾部的标志：java.io.EOFException。

三、字符编码

常见字符编码：

编码	说明
ISO-8859-1	收录除ASCII外，还包括西欧、希腊语、泰语、阿拉伯语、希伯来语对应的文字符号。
UTF-8	针对Unicode的可变长度字符编码。
GB2312	简体中文。
GBK	简体中文、扩充。
BIG5	台湾，繁体中文。

注：当编码方式和解码方式不一致时，会出现乱码。

四、字符流【重点】

4.1 字符抽象类

Reader：字符输入流

- public int read(){}。
- public int read(char[] c){}。
- public int read(char[] b,int off,int len){}。

Writer：字符输出流

- public void write(int n){}。
- public void write(String str){}。
- public void write(char[] c){}。

4.2 字符节点流

FileWriter：

- public void write(String str) 。
- 一次写多个字符，将b数组中所有字符，写入输出流。

FileReader：

- public int read(char[] c) 。
- 从流中读取多个字符，将读到内容存入c数组，返回实际读到的字符数；如果达到文件的尾部，则返回-1。

案例演示：

```
public class TestFileReader {
  public static void main(String[] args) throws Exception{
    //1创建FileReader 文件字符输入流
    FileReader fr=new FileReader("d:\\hello.txt");
    //2读取
    //2.1单个字符读取
    //int data=0;
    //while((data=fr.read())!=-1) {//读取一个字符
    //  System.out.print((char)data);
    //}
    char[] buf=new char[1024];
    int count=0;
    while((count=fr.read(buf))!=-1) {
      System.out.println(new String(buf, 0, count));
    }

    //3关闭
    fr.close();
  }
}
```

```
public class TestFileWriter {
```

```
  public static void main(String[] args) throws Exception {
    //1创建FileWriter对象
    FileWriter fw=new FileWriter("d:\\write.txt");
    //2写入
    for(int i=0;i<10;i++) {
      fw.write("java是世界上最好的语言\r\n");
      fw.flush();
    }
    //3关闭
    fw.close();
    System.out.println("执行完毕");
  }
}
```

4.3 字符缓冲流

> 缓冲流：BufferedWriter/BufferedReader
>
> - 支持输入换行符。
> - 可一次写一行、读一行。

> 案例演示：

```
public class TestBufferedReader {
  public static void main(String[] args) throws Exception{
    //1创建缓冲流
    FileReader fr=new FileReader("d:\\write.txt");
    BufferedReader br=new BufferedReader(fr);
    //2读取
    //2.1第一种方式
    //char[] buf=new char[1024];
    //int count=0;
    //while((count=br.read(buf))!=-1) {
    //  System.out.print(new String(buf,0,count));
    //}
    //2.2第二种方式，一行一行的读取
    String line=null;
    while((line=br.readLine())!=null) {
      System.out.println(line);
    }

    //3关闭
    br.close();
  }
}
```

```
public class TestBufferedWriter {
  public static void main(String[] args) throws Exception{
    //1创建BufferedWriter对象
    FileWriter fw=new FileWriter("d:\\buffer.txt");
    BufferedWriter bw=new BufferedWriter(fw);
    //2写入
    for(int i=0;i<10;i++) {
      bw.write("好好学习，天天向上");
      bw.newLine();//写入一个换行符 windows \r\n  linux  \n
      bw.flush();
    }
    //3关闭
    bw.close();
    System.out.println("执行完毕");

  }
}
```

4.4 打印流

> PrintWriter：
>
> - 封装了print() / println()方法，支持写入后换行。
> - 支持数据原样打印。

> 案例演示：

```
public class TestPrintWriter {
  public static void main(String[] args) throws Exception {
    //1创建打印流
    PrintWriter pw=new PrintWriter("d:\\print.txt");
    //2打印
    pw.println(97);
    pw.println(true);
    pw.println(3.14);
    pw.println('a');
    //3关闭
    pw.close();
```

```
    System.out.println("执行完毕");
  }
}
```

4.5 转换流

> 转换流：InputStreamReader/OutputStreamWriter
>
> - 可将字节流转换为字符流。
> - 可设置字符的编码方式。

> 案例演示：

```
public class TestInputStreamReader {
  public static void main(String[] args) throws Exception {
    //1创建InputStreamReader对象
    FileInputStream fis=new FileInputStream("d:\\write.txt");
    InputStreamReader isr=new InputStreamReader(fis, "gbk");
    //2读取文件
    int data=0;
    while((data=isr.read())!=-1) {
      System.out.print((char)data);
    }
    //3关闭
    isr.close();
  }
}
```

```
public class TestOutputStreamWriter {
  public static void main(String[] args) throws Exception{
    //1创建OutputStreamWriter
    FileOutputStream fos=new FileOutputStream("d:\\info.txt");
    OutputStreamWriter osw=new OutputStreamWriter(fos, "utf-8");
    //2写入
    for(int i=0;i<10;i++) {
      osw.write("我爱北京，我爱故乡\r\n");
      osw.flush();
    }
    //3关闭
    osw.close();
    System.out.println("执行成功");
  }
}
```

五、File、FileFilter

5.1 File类

> 概念：代表物理盘符中的一个文件或者文件夹。

> 常见方法：

方法名	描述
createNewFile()	创建一个新文件。
mkdir()	创建一个新目录。
delete()	删除文件或空目录。
exists()	判断File对象所对象所代表的对象是否存在。
getAbsolutePath()	获取文件的绝对路径。
getName()	取得名字。
getParent()	获取文件/目录所在的目录。
isDirectory()	是否是目录。
isFile()	是否是文件。
length()	获得文件的长度。
listFiles()	列出目录中的所有内容。
renameTo()	修改文件名为。

> 案例演示：

```
public class TestFile {
  public static void main(String[] args) throws Exception {
    //separator();
    //fileOpe();
    directoryOpe();
  }
  // (1) 分隔符
  public static void separator() {
    System.out.println("路径分隔符"+File.pathSeparator);
    System.out.println("名称分隔符"+File.separator);
  }
  // (2) 文件操作
  public static void fileOpe() throws Exception {
    //1创建文件 createNewFile()
    File file=new File("d:\\file.txt");
    //System.out.println(file.toString());
    if(!file.exists()) {
      boolean b=file.createNewFile();
      System.out.println("创建结果:"+b);
    }
    //2删除文件
    //2.1直接删除
    //System.out.println("删除结果:"+file.delete());
    //2.2使用jvm退出时删除
    //file.deleteOnExit();
    //Thread.sleep(5000);

    //3获取文件信息
    System.out.println("获取文件的绝对路径:"+file.getAbsolutePath());
    System.out.println("获取路径:"+file.getPath());
    System.out.println("获取文件名称:"+file.getName());
    System.out.println("获取父目录:"+file.getParent());
    System.out.println("获取文件长度:"+file.length());
    System.out.println("文件创建时间:"+new Date(file.lastModified()).toLocaleString());


    //4判断
    System.out.println("是否可写:"+file.canWrite());
    System.out.println("是否时文件:"+file.isFile());
    System.out.println("是否隐藏:"+file.isHidden());

  }

  // (3) 文件夹操作
  public static void directoryOpe() throws Exception{
    //1 创建文件夹
    File dir=new File("d:\\aaa\\bbb\\ccc");
    System.out.println(dir.toString());
    if(!dir.exists()) {
      //dir.mkdir();//只能创建单级目录
      System.out.println("创建结果:"+dir.mkdirs());//创建多级目录
    }

    //2 删除文件夹
    //2.1直接删除(注意删除空目录)
    //System.out.println("删除结果:"+dir.delete());
    //2.2使用jvm删除
    //dir.deleteOnExit();
    //Thread.sleep(5000);
    //3获取文件夹信息
    System.out.println("获取绝对路径: "+dir.getAbsolutePath());
    System.out.println("获取路径:"+dir.getPath());
    System.out.println("获取文件夹名称: "+dir.getName());
    System.out.println("获取父目录: "+dir.getParent());
    System.out.println("获取创建时间:"+new Date(dir.lastModified()).toLocaleString());

    //4判断
    System.out.println("是否时文件夹:"+dir.isDirectory());
    System.out.println("是否时隐藏: "+dir.isHidden());

    //5遍历文件夹
    File dir2=new File("d:\\图片");
    String[] files=dir2.list();
    System.out.println("---------------------------------");
    for (String string : files) {
      System.out.println(string);
    }

  }

}
```

5.2 FileFilter接口

> FileFilter：文件过滤器接口
>
> - boolean accept(File pathname)。
> - 当调用File类中的listFiles()方法时，支持传入FileFilter接口接口实现类，对获取文件进行过滤，只有满足条件的文件的才可出现在

> listFiles()的返回值中。

> 案例演示：过滤所有的.jpg图片。

```
public class TestFileFilter{
  public static void main(String[] args){
    File dir=new File("d:\\图片");
    File[] files2=dir.listFiles(new FileFilter() {

      @Override
      public boolean accept(File pathname) {
        if(pathname.getName().endsWith(".jpg")) {
          return true;
        }
        return false;
      }
    });
    for (File file : files2) {
      System.out.println(file.getName());
    }
  }
}
```

六、Properties实现流操作

> Properties：属性集合。

> 特点：
>
> - 存储属性名和属性值。
> - 属性名和属性值都是字符串类型。
> - 没有泛型。
> - 和流有关。

> 案例演示：Properties实现流操作。

```
public class TestProperties {
  public static void main(String[] args) throws Exception {
    //1创建集合
    Properties properties=new Properties();
    //2添加数据
    properties.setProperty("username", "zhangsan");
    properties.setProperty("age", "20");
    System.out.println(properties.toString());
    //3遍历
    //3.1-----keySet----略
    //3.2-----entrySet----略
    //3.3-----stringPropertyNames()---
    Set<String> pronames=properties.stringPropertyNames();
    for (String pro : pronames) {
      System.out.println(pro+"====="+properties.getProperty(pro));
    }
    //4和流有关的方法
    //----------1、list方法---------
    PrintWriter pw=new PrintWriter("d:\\print.txt");
    properties.list(pw);
    pw.close();

    //----------2、store方法 保存-----------
    FileOutputStream fos=new FileOutputStream("d:\\store.properties");
    properties.store(fos, "注释");
    fos.close();

    //----------3、load方法 加载-------------
    Properties properties2=new Properties();
    FileInputStream fis=new FileInputStream("d:\\store.properties");
    properties2.load(fis);
    fis.close();
    System.out.println(properties2.toString());

  }
}
```