In Memory IO Streams

Introducing I/O Streams on In-Memory Structures

The JAVA I/O API also gives classes to access the content of in-memory structures, namely arrays of characters or bytes, and strings of characters. There are several use cases where this feature is very handy.

Certain file formats (this is the case for the JPEG file format) require a special field at the beginning of the file, that gives the length of certain portions or fields of the file. There are cases where it is not possible to compute these portions in advance. Think of compressed data: computing the size of set of 100 integers is easy, but computing it once it has been gzipped is much harder. With the right class, you can create this gzipped stream in an array of bytes, and simply get the number of the written bytes. This example is covered at the end of this section.

Reading and Writing Arrays of Characters

The `CharArrayReader` and `CharArrayWriter` are both wrapping an array of `char`, specified at the construction of these classes. They are both extensions

of `Reader` and `Writer` (respectively), and do not add any methods to these classes.

Reading and Writing Strings of Characters

The `StringReader` class is also an extension of the abstract class `Reader`. It is built on a `String`, passed as an argument to its constructor. It does not add any method to the `Reader` class.

The `StringWriter` is a little different. It wraps an internal `StringBuffer` and can append characters to it. You can then get this `StringBuffer` by calling one of the two following methods.

1. `getBuffer()`: returns the internal `StringBuffer`. No defensive copy is made here.
2. `toString()`: returns a string of characters built by calling the `toString()` method of the internal `StringBuffer`.

Reading and Writing Arrays of Bytes

Two classes are available to read and write bytes in arrays: `ByteArrayInputStream` and `ByteArrayOutputStream`.

The first one allows you to read the content of a byte array as an `InputStream`, provided as an argument to the constructor of this class.

The second one allows you to write bytes to a byte array. You can fix the initial size of this array, and it will grow automatically if it becomes full. Once the bytes have been written, you can get the content of this array in different ways.

1. `size()` gives you the number of bytes contained in this array.

2. `toString()` returns the content of the array as a string of characters. This method can take a `CharSet` as an argument to decode these bytes correctly.
3. `toBytes()` returns a copy of the internal array of this `ByteArrayOutputStream`.

Last update: *January 25, 2023*

Decorating IO Streams	←	**In Memory IO Streams**	→	That's the end of the series!

Home > Tutorials > The Java I/O API > File Operations Basics > In Memory IO Streams