iHarder.net

Base64

Something Sun Should Have Included Long Ago

Download now! (v2.2.1 48KB)

Alert me to new releases...

Now supports GZip-(de)compressing data before/after encoding!

This is a **Public Domain** Java class providing **very fast** Base64 encoding and decoding in the form of convenience methods and input/output streams.

There are other Base64 utilities on the Internet, some part of proprietary packages, some with various open source licenses. In any event, I hope with one or more of these Base64 tools, you won't have to write your own like I did.

Thanks to Brian Burton for providing this Base64Test.java test class for use with JUnit.org.

Changes

- v2.2.1 - Fixed bug using URL_SAFE and ORDERED encodings. Fixed bug when using very small files (~< 40 bytes).

- v2.2 - Added some helper methods for encoding/decoding directly from one file to the next. Also added a main() method to support command line encoding/decoding from one file to the next. Also added these Base64 dialects:
  1. The default is RFC3548 format.
  2. Calling Base64.setFormat(Base64.BASE64_FORMAT.URLSAFE_FORMAT) generates URL and file name friendly format as described in Section 4 of RFC3548. http://www.faqs.org/rfcs/rfc3548.html
  3. Calling Base64.setFormat(Base64.BASE64_FORMAT.ORDERED_FORMAT) generates URL and file name friendly format that preserves lexical ordering as described in http://www.faqs.org/qa/rfcc-1940.html

  Special thanks to Jim Kellerman at http://www.powerset.com/ for contributing the new Base64 dialects.
- v2.1 - Cleaned up javadoc comments and unused variables and methods. Added some convenience methods for reading and writing to and from files.
- v2.0.2 - Now specifies UTF-8 encoding in places where the code fails on systems with other encodings (like EBCDIC).
- v2.0.1 - Fixed an error when decoding a single byte, that is, when the encoded data was a single byte.
- v2.0 - I got rid of methods that used booleans to set options. Now everything is more consolidated and cleaner. The code now detects when data that's being decoded is gzip-compressed and will decompress it automatically. Generally things are cleaner. You'll probably have to change some method calls that you were making to support the new options format (`int`s that you "OR" together).
- v1.5.1 - Fixed bug when decompressing and decoding to a byte[] using `decode( String s, boolean gzipCompressed )`. Added the ability to "suspend" encoding in the Output Stream so you can turn on and off the encoding if you need to embed base64 data in an otherwise "normal" stream (like an XML file). *This has not been fully tested, so please alert me to bugs.*
- v1.5 - Output stream pases on flush() command but doesn't do anything itself. This helps when using GZIP streams. Added the ability to **GZip-compress objects** before encoding them.

- v1.4 - Added some helper methods for reading and writing to/from files.
- v1.3.6 - Fixed OutputStream.flush() so that 'position' is reset.
- v1.3.5 - Added flag to turn on and off line breaks. Fixed bug in input stream where last buffer being read, if not completely full, was not returned.
- v1.3.4 - Fixed when *Improperly padded base64 stream* exception was incorrectly thrown.
- A bug has been fixed that kept I/O streams from working at all, really.
- A bug has been fixed affecting you if you use the Base64.InputStream to encode data.
- A bug has been fixed where if you specified an offset when encoding an array of bytes, the offset was ignored.

Examples

The easiest way to convert some data is with the convenience methods:

```
String result1 = Base64.encodeObject( mySerializableObject );
String result2 = Base64.encodeBytes( new byte[]{ 3, 34, 116, 9 } );
```

Or you can use the very efficient streams:

```
OutputStream out = new Base64.OutputStream(
                    new FileOutputStream( "out.txt" ) );
// Go on about your outputting...
// ...

InputStream in = new Base64.InputStream(
                  new FileInputStream( "in.txt" ) );
// Go on about your inputting...
// ...
```

There are defaults (OutputStream encodes, InputStream decodes), but you can easily override that:

```
OutputStream out = new Base64.OutputStream(
                     new FileOutputStream( "out.txt" ), Base64.DECODE );
// Go on about your outputting...
// ...
```

A Note About Public Domain

I have released this software into the Public Domain. That means you can

do whatever you want with it. Really. You don't have to match it up with any other open source license &em; just use it. You can rename the files, move the Java packages, whatever you want. If your lawyers say you have to have a license, contact me, and I'll make a special release to you under whatever reasonable license you desire: MIT, BSD, GPL, whatever.