CompressRoadmap

Compress Roadmap

About this page

Compress has seen several releases of the 1.x series. While the factory and stream APIs have proven to be useful there are pieces that "don't feel right" and cannot be changed without breaking backwards compatibility. Also Compress' API has been designed for Java 1.4 and could benefit from generics and enums.

The idea of a 2.0 release that is allowed to break backwards compatibility has come up more than once over a span of five years or even longer. This page will gather requirements and design ideas in the hope that a real implementation will be created based on the existing code.

A starting ground for some design ideas is the compress-2.0 branch https://git-wip-us.apache.org/repos/asf?p=commons-compress.git;a=tree;h=refs/heads/compress-2.0;hb=refs/heads/compress-2.0 - nothing carved into stone, yet. Feedback, ideas and corrections more than welcome.

General

- SETTLED: Compress 2.0 will require Java8 at compile and run time.

- external dependencies?
  we have copied or reinvented some code from IO - do we want to keep our copy or use a dependency?
- some general cleanup
  we have encoding code in the zip packages that gets used in tar and other archived classes. We should make a run for extracting common code into utilities.
  Some names could be better like the constants in ZipMethod.
- make the factories configurable - i.e. allow third parties to register new formats without changing Compress ServiceProvider?
- common solutions for streaming
  pack200 has to use a temporary files or hold the result in memory as it uses an API that converts a stream with a single call. The archive classes that need random access can currently only work on files. This will change for 1.13 with support for SeekableByteChannel
- read-only support
  The list of formats we can read but not write seems to be growing. Do we want to add some sort of meta-data for a format we can query to know whether it supports writing? Might be part of the "make factories configurable" solution.
- events for certain stages of (un)archiving/(un)compressing?
  This might be used for progress bars or similar stuff at a higher level. COMPRESS-207
- a common solution for things that are extensible inside a given format like an API that allows third parties to implement und use compression /encryption methods without modifying Compress' codebase.
  This is COMPRESS-143 for ZIP but also applies to 7z or for extra fields inside the ZIP format.

Archivers

- unify common stuff in ArchiveEntry
  this includes extracting a common representation for modes/permission (COMPRESS-136) but doesn't need to stop there. There is a discussion thread that is more than three years old with ideas about this: http://markmail.org/thread/fsxtzs3vsepycu25
- embrace generics at least for the `getNextEntry`/`putArchiveEntry` methods
- streaming vs random-access
  For some archivers we don't have streams as their formats really require random access to work properly (7z) for others there is a stream and a diferent class for random access (zip) which is superior. Should we generalize this in some way like having a factory and a common API for the random access based classes? See also the "general" bullet point on this.

JAR

- stop extending the zip stuff but do something useful for JAR archives like providing access to the manifest. COMPRESS-18

ZIP

- try to share more code between ZipFile and InputStream

Compressors

- provide byte[] based Compressors/Decompressors as an alternative to streams
  COMPRESS-134

...