

CompressImplementationDetails

This page is obsolete and kept for its historical value

Compress Implementation Details

About this page

There has been some discussion in the past about improving compress and the need to get it out of the sandbox. Some of this has been discussed starting with this message: http://mail-archives.apache.org/mod_mbox/commons-dev/200708.mbox/%3c46CE592C.50200@wso2.com%3e

Compress is considered to be complex in its implementation; it seems to be difficult to contribute new Compressors/Archivers. This page shall give a basis for further design discussions and help people to jump into the implementation.

"Modules"

Compress started with implementations to create zip-, tar- and bzip2-files (called providers from now on). In 2006 a common interface for these implementations has been added. The common interface can be found in `org.apache.commons.compress` and the providers in `org.apache.commons.compress.archivers` and `org.apache.commons.compress.compressors`.

As you can see, the providers have been splitted up into compressor (f.e. BZip2) and archivers (tar and zip, although zip provides compression too). At the moment there are redundant provider packages in the root package, which is by fault.

Providers

The providers are quite long part of these projects; they have been extracted from other Apache projects and didn't change for a long time. At the moment everything works quite well with a few Jira issues. The code should be more tidy and could have some more documentation. Sometimes code can be improved.

Common Interface

The Common Interface consists of a factory, namely [CompressorFactory](#) and [ArchiveFactory](#), which return Compressor or Archives. In fact with the introduction of the common interface each provider has become a new class which implements one of these interfaces, Compressor or Archive. For example, there is now a BZip2Compressor.

There were many common operations for Compressors and Archives, so [AbstractArchive](#) and [AbstractCompressor](#) has been written, which provide these operations. One can now simply extend f.e. [AbstractCompressor](#) and gain these functionality. With doing so, a new provider has just to implement a few methods; in case of a compressor, there should be the following:

- `compressTo(FileInputStream inputStream, FileOutputStream outputStream)` throws [CompressException](#)
- `decompressTo(FileInputStream input, FileOutputStream outputStream)` throws [CompressException](#)

and also some Methods like `getHeader`, which is used to identify a file by its byte-header, `getName` or the `getDefaultExtension`, which is used for creating new files (in case of a Zip-File, it "zip").

Other classes

There are just a few other classes, like [ArchiveException](#) or [CompressException](#) (selfspeaking). [CompressUtils](#) provides some kind of copy-function. [PackableObject](#), the mother of [AbstractArchive](#) and [AbstractCompressor](#) provides the "Identify by Header" Functionity. [ArchiveEntry](#) is just a representation of an Entry in an Archive.

Ideas for enhancing the design

TODO