Zip File DataFormat

Zip File

Available since Camel 2.11.0

The Zip File Data Format is a message compression and de-compression format. Messages can be marshalled (compressed) to Zip files containing a single entry, and Zip files containing a single entry can be unmarshalled (decompressed) to the original file contents. This data format supports ZIP64, as long as Java 7 or later is being used.

Since Camel 2.12.3 there is also a aggregation strategy that can aggregate multiple messages into a single Zip file.

Marshal

In this example we marshal a regular text/XML payload to a compressed payload using Zip file compression, and send it to an ActiveMQ queue called MY_QUEUE.

```
from("direct:start").marshal().zipFile().to("activemq:queue:MY_QUEUE");
```

The name of the Zip entry inside the created Zip file is based on the incoming `CamelFileName` message header, which is the standard message header used by the file component. Additionally, the outgoing `CamelFileName` message header is automatically set to the value of the incoming `CamelFileName` message header, with the ".zip" suffix. So for example, if the following route finds a file named "test.txt" in the input directory, the output will be a Zip file named "test.txt.zip" containing a single Zip entry named "test.txt":

```
from("file:input/directory?antInclude=*/.txt").marshal().zipFile().to("file:output/directory");
```

If there is no incoming `CamelFileName` message header (for example, if the file component is not the consumer), then the message ID is used by default, and since the message ID is normally a unique generated ID, you will end up with filenames like `ID-MACHINENAME-2443-1211718892437-1-0.zip`. If you want to override this behavior, then you can set the value of the `CamelFileName` header explicitly in your route:

```
from("direct:start").setHeader(Exchange.FILE_NAME, constant("report.txt")).marshal().zipFile().to("file:output
/directory");
```

This route would result in a Zip file named "report.txt.zip" in the output directory, containing a single Zip entry named "report.txt".

Unmarshal

In this example we unmarshal a Zip file payload from an ActiveMQ queue called MY_QUEUE to its original format, and forward it for processing to the `UnZippedMessageProcessor`.

```
from("activemq:queue:MY_QUEUE").unmarshal().zipFile().process(new UnZippedMessageProcessor());
```

If the zip file has more then one entry, the usingIterator option of ZipFileDataFormat to be true, and you can use splitter to do the further work.

```
  ZipFileDataFormat zipFile = new ZipFileDataFormat();
  zipFile.setUsingIterator(true);
  from("file:src/test/resources/org/apache/camel/dataformat/zipfile/?consumer.delay=1000&noop=true")
    .unmarshal(zipFile)
    .split(body(Iterator.class))
        .streaming()
          .process(new UnZippedMessageProcessor())
    .end();
```

Or you can use the ZipSplitter as an expression for splitter directly like this

```
   from("file:src/test/resources/org/apache/camel/dataformat/zipfile?consumer.delay=1000&noop=true")
     .split(new ZipSplitter())
        .streaming()
        .process(new UnZippedMessageProcessor())
     .end();
```

Aggregate

Available since Camel 2.12.3

Please note that this aggregation strategy requires eager completion check to work properly.

In this example we aggregate all text files found in the input directory into a single Zip file that is stored in the output directory.

```
   from("file:input/directory?antInclude=*/.txt")
     .aggregate(new ZipAggregationStrategy())
       .constant(true)
       .completionFromBatchConsumer()
       .eagerCheckCompletion()
   .to("file:output/directory");
```

The outgoing `CamelFileName` message header is created using java.io.File.createTempFile, with the ".zip" suffix. If you want to override this behavior, then you can set the value of the `CamelFileName` header explicitly in your route:

```
   from("file:input/directory?antInclude=*/.txt")
     .aggregate(new ZipAggregationStrategy())
       .constant(true)
       .completionFromBatchConsumer()
       .eagerCheckCompletion()
     .setHeader(Exchange.FILE_NAME, constant("reports.zip"))
   .to("file:output/directory");
```

Dependencies

To use Zip files in your camel routes you need to add a dependency on **camel-zipfile** which implements this data format.

If you use Maven you can just add the following to your `pom.xml`, substituting the version number for the latest & greatest release (see the download page for the latest versions).

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-zipfile</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```