**Camel 2.0 onwards default error handler**

In Camel 2.0 onwards a global [DefaultErrorHandler](http://camel.apache.org/defaulterrorhandler.html) is set up as the [Error Handler](http://camel.apache.org/error-handler.html) by default. It's configured as:

* no redeliveries
* no dead letter queue
* if the exchange failed an exception is thrown and propagated back to the original caller wrapped in a RuntimeCamelException.

……

To keep things simple we first look at the basic concept how Camel orchestrates the redelivery attempt. At any given node in the route graph Camel intercepts the current Exchange being routed and wraps it with the [Error Handler](http://camel.apache.org/error-handler.html). This ensures that the [Error Handler](http://camel.apache.org/error-handler.html) can kick in, just as the AOP around concept. If the exchange can be routed without any problems then it's forwarded to the next node in the route graph, **But** if there was an exception thrown, then the [Error Handler](http://camel.apache.org/error-handler.html) kicks in and decides what to do.

An example illustrating this:

|  |
| --- |
| errorHandler(deadLetterChannel("jms:queue:dead"));    from("seda:newOrder")     .to("bean:validateOrder")     .to("bean:storeOrder")     .to("bean:confirmOrder"); |

In this route we have 3 nodes (the dots) where the [Error Handler](http://camel.apache.org/error-handler.html) is watching us (The AOP around stuff). So when an order arrives on the seda queue we consume it and send it to the validateOrder bean. In case the validation bean processed ok, we move on to the next node. In case the storeOrder bean failed and throws an exception it's caught by the [Dead Letter Channel](http://camel.apache.org/dead-letter-channel.html) that decides what to do next. Either it does a:

* redeliver
* or move it to dead letter queue

It will continue to do redeliveries based on the policy configured. By default [Dead Letter Channel](http://camel.apache.org/dead-letter-channel.html) will attempt at most 6 redeliveries with 1 second delay. So if the storeOrder bean did succeed at the 3rd attempt the routing will continue to the next node the confirmOrder bean. In case all redeliveries failed the Exchange is regarded as failed and is moved to the dead letter queue and the processing of this exchange stops. By default the dead letter queue is just a ERROR logger.

<http://camel.apache.org/error-handling-in-camel.html>

### Configuring RedeliveryPolicy (redeliver options)

[RedeliveryPolicy](http://camel.apache.org/maven/current/camel-core/apidocs/org/apache/camel/processor/RedeliveryPolicy.html) requires to use the [Dead Letter Channel](http://camel.apache.org/dead-letter-channel.html) as the [Error Handler](http://camel.apache.org/error-handler.html). Dead Letter Channel supports attempting to redeliver the message exchange a number of times before sending it to a dead letter endpoint. See [Dead Letter Channel](http://camel.apache.org/dead-letter-channel.html) for further information about redeliver and which redeliver options exists.

No redelivery is default for onException

By default any [Exception Clause](http://camel.apache.org/exception-clause.html) will **not** redeliver! (as it sets the maximumRedeliveries option to 0).

<http://camel.apache.org/exception-clause.html>

## Dead Letter Channel

The **DefaultErrorHandler** does very little: it ends the Exchange immediately and propagates the thrown Exception back to the caller.(the route start endpoint)

The **DeadLetterChannel** lets you control behaviors including redelivery, whether to propagate the thrown Exception to the caller (the **handled** option), and where the (failed) Exchange should now be routed to.

### Redelivery

It is common for a temporary outage or database deadlock to cause a message to fail to process; but the chances are if its tried a few more times with some time delay then it will complete fine. So we typically wish to use some kind of redelivery policy to decide how many times to try redeliver a message and how long to wait before redelivery attempts.

The [RedeliveryPolicy](http://camel.apache.org/maven/current/camel-core/apidocs/org/apache/camel/processor/RedeliveryPolicy.html) defines how the message is to be redelivered. You can customize things like

* The number of times a message is attempted to be redelivered before it is considered a failure and sent to the dead letter channel.
* The initial redelivery timeout.
* Whether or not exponential backoff is used, i.e., the time between retries increases using a backoff multiplier.
* Whether to use collision avoidance to add some randomness to the timings.
* Delay pattern (see below for details).
* **Camel 2.11:** Whether to allow redelivery during stopping/shutdown.

Once all attempts at redelivering the message fails then the message is forwarded to the dead letter queue.

<http://camel.apache.org/dead-letter-channel.html>