Delayer

Delayer

The Delayer Pattern allows you to delay the delivery of messages to some destination.

Delayer
The expression is a value in millis to wait from the current time, so the expression should just be `3000`.
However you can use a long value for a fixed value to indicate the delay in millis.
See the Spring DSL samples for Delayer.
Using Delayer in Java DSL
See this ticket: https://issues.apache.org/jira/browse/CAMEL-2654

Options

confluenceTableSmall

Name	Default Value	Description
`asyncDelayed`	`false`	**Camel 2.4:** If enabled then delayed messages happens asynchronously using a scheduled thread pool.
`executorServiceRef`		**Camel 2.4:** Refers to a custom Thread Pool to be used if `asyncDelay` has been enabled.
`callerRunsWhenRejected`	`true`	**Camel 2.4:** Is used if `asyncDelayed` was enabled. This controls if the caller thread should execute the task if the thread pool rejected the task.

Using the Fluent Builders

The example below will delay all messages received on **seda:b** 1 second before sending them to **mock:result**.

{snippet:id=ex2|lang=java|url=camel/trunk/camel-core/src/test/java/org/apache/camel/processor/DelayerTest.java}

You can just delay things a fixed amount of time from the point at which the delayer receives the message. For example to delay things 2 seconds

delayer(2000)

The above assume that the delivery order is maintained and that the messages are delivered in delay order. If you want to reorder the messages based on delivery time, you can use the Resequencer with this pattern. For example

from("activemq:someQueue").resequencer(header("MyDeliveryTime")).delay("MyRedeliveryTime").to("activemq:aDelayedQueue");

You can of course use many different Expression languages such as XPath, XQuery, SQL or various Scripting Languages. For example to delay the message for the time period specified in the header, use the following syntax:

from("activemq:someQueue").delay(header("delayValue")).to("activemq:aDelayedQueue");

And to delay processing using the Simple language you can use the following DSL:

from("activemq:someQueue").delay(simple("${body.delayProperty}")).to("activemq:aDelayedQueue");

Spring DSL

The sample below demonstrates the delay in Spring DSL:

{snippet:id=example|lang=xml|url=camel/trunk/components/camel-spring/src/test/resources/org/apache/camel/spring/processor/delayer.xml}

For further examples of this pattern in use you could look at the junit test case

Asynchronous delaying

Available as of Camel 2.4

You can let the Delayer use non blocking asynchronous delaying, which means Camel will use a scheduler to schedule a task to be executed in the future. The task will then continue routing. This allows the caller thread to not block and be able to service other messages etc.

From Java DSL

You use the `asyncDelayed()` to enable the async behavior.

from("activemq:queue:foo").delay(1000).asyncDelayed().to("activemq:aDelayedQueue");

From Spring XML

You use the `asyncDelayed="true"` attribute to enable the async behavior.

xml<route> <from uri="activemq:queue:foo"/> <delay asyncDelayed="true"> <constant>1000</constant> </delay> <to uri="activemq:aDealyedQueue"/> < /route>

Creating a custom delay

You can use an expression to determine when to send a message using something like this

from("activemq:foo"). delay().method("someBean", "computeDelay"). to("activemq:bar");

then the bean would look like this...

public class SomeBean { public long computeDelay() { long delay = 0; // use java code to compute a delay value in millis return delay; } }

Using This Pattern

See Also

- Delay Interceptor