# Loop

## Loop

The Loop allows for processing a message a number of times, possibly in a different way for each iteration. Useful mostly during testing.

Default mode
Notice by default the loop uses the same exchange throughout the looping. So the result from the previous iteration will be used for the next (eg Pipes and Filters). From **Camel 2.8** onwards you can enable copy mode instead. See the options table for more details.

### Options

confluenceTableSmall

| Name | Default Value | Description |
|---|---|---|
| `copy` | `false` | **Camel 2.8:** Whether or not copy mode is used. If `false` then the same Exchange will be used for each iteration. So the result from the previous iteration will be *visible* for the next iteration. Instead you can enable copy mode, and then each iteration *restarts* with a fresh copy of the input Exchange. |
| doWhile | | **Camel 2.17:** Enables the while loop that loops until the predicate evaluates to false or null. |

### Exchange properties

For each iteration two properties are set on the `Exchange`. Processors can rely on these properties to process the Message in different ways.

| Property | Description |
|---|---|
| `CamelLoopSize` | Total number of loops. This is not available if running the loop in while loop mode. |
| `CamelLoopIndex` | Index of the current iteration (0 based) |

### Examples

The following example shows how to take a request from the **direct:x** endpoint, then send the message repetitively to **mock:result**. The number of times the message is sent is either passed as an argument to `loop()`, or determined at runtime by evaluating an expression. The expression **must** evaluate to an `int`, otherwise a `RuntimeCamelException` is thrown.

**Using the Fluent Builders**

Pass loop count as an argument{snippet:id=ex1|lang=java|url=camel/trunk/camel-core/src/test/java/org/apache/camel/processor/LoopTest.java}Use expression to determine loop count{snippet:id=ex2|lang=java|url=camel/trunk/camel-core/src/test/java/org/apache/camel/processor/LoopTest.java}Use expression to determine loop count{snippet:id=ex3|lang=java|url=camel/trunk/camel-core/src/test/java/org/apache/camel/processor/LoopTest.java}**Using the Spring XML Extensions**

Pass loop count as an argument{snippet:id=ex1|lang=xml|url=camel/trunk/components/camel-spring/src/test/resources/org/apache/camel/spring/processor/loop.xml}Use expression to determine loop count{snippet:id=ex2|lang=xml|url=camel/trunk/components/camel-spring/src/test/resources/org/apache/camel/spring/processor/loop.xml}For further examples of this pattern in use you could look at one of the junit test case

### Using copy mode

**Available as of Camel 2.8**

Now suppose we send a message to "direct:start" endpoint containing the letter A.
The output of processing this route will be that, each "mock:loop" endpoint will receive "AB" as message.{snippet:id=e1|lang=java|url=camel/trunk/camel-core/src/test/java/org/apache/camel/processor/LoopCopyTest.java}However if we do **not** enable copy mode then "mock:loop" will receive "AB", "ABB", "ABBB", etc. messages.{snippet:id=e1|lang=java|url=camel/trunk/camel-core/src/test/java/org/apache/camel/processor/LoopNoCopyTest.java}The equivalent example in XML DSL in copy mode is as follows:{snippet:id=e1|lang=xml|url=camel/trunk/components/camel-spring/src/test/resources/org/apache/camel/spring/processor/SpringLoopCopyTest.xml}

### Using while mode

**Available as of Camel 2.17**

The loop can act like a while loop that loops until the expression evaluates to false or null.

For example the route below loops while the length of the message body is 5 or less characters. Notice that the DSL uses **loopDoWhile**.

from("direct:start") .loopDoWhile(simple("${body.length} <= 5")) .to("mock:loop") .transform(body().append("A")) .end() .to("mock:result");

And the same example in XML:

xml <route> <from uri="direct:start"/> <loop doWhile="true"> <simple>${body.length} &lt;= 5</simple> <to uri="mock:loop"/> <transform> <simple>A${body} </simple> </transform> </loop> <to uri="mock:result"/> </route>

Notice in XML that the while loop is turned on using the **doWhile** attribute.

Using This Pattern