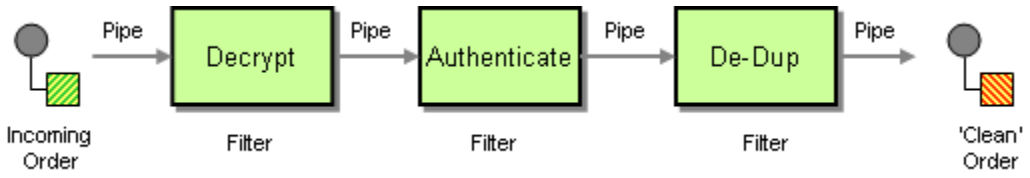
Pipes and Filters

Pipes and Filters

Camel supports the Pipes and Filters from the EIP patterns in various ways.



With Camel you can split your processing across multiple independent Endpoint instances which can then be chained together.

Using Routing Logic

You can create pipelines of logic using multiple Endpoint or Message Translator instances as follows{snippet:id=example|lang=java|url=camel/trunk/camel-core/src/test/java/org/apache/camel/processor/PipelineTest.java}Though pipeline is the default mode of operation when you specify multiple outputs in Camel. The opposite to pipeline is multicast; which fires the same message into each of its outputs. (See the example below).

In Spring XML you can use the **`<pipeline/>`** element

xml<route> <from uri="activemq:SomeQueue"/> <pipeline> <bean ref="foo"/> <bean ref="bar"/> <to uri="activemq:OutputQueue"/> </pipeline> </route>

In the above the pipeline element is actually unnecessary, you could use this:

xml<route> <from uri="activemq:SomeQueue"/> <bean ref="foo"/> <bean ref="bar"/> <to uri="activemq:OutputQueue"/> </route>

which is a bit more explicit.

However if you wish to use **`<multicast/>`** to avoid a pipeline - to send the same message into multiple pipelines - then the **`<pipeline/>`** element comes into its own:

xml<route> <from uri="activemq:SomeQueue"/> <multicast> <pipeline> <bean ref="something"/> <to uri="log:Something"/> </pipeline> <pipeline> <bean ref="foo"/> <bean ref="bar"/> <to uri="activemq:OutputQueue"/> </pipeline> </multicast> </route>

In the above example we are routing from a single Endpoint to a list of different endpoints specified using URIs. If you find the above a bit confusing, try reading about the Architecture or try the Examples

Using This Pattern