**Processor**

The [Processor](http://camel.apache.org/maven/current/camel-core/apidocs/org/apache/camel/Processor.html) interface is used to implement consumers of message exchanges or to implement a [Message Translator](http://camel.apache.org/message-translator.html)

**Using a processor in a route**

Once you have written a class which implements processor like this...

|  |
| --- |
| public class MyProcessor implements Processor {    public void process(Exchange exchange) throws Exception {      // do something...    }  } |

You can then easily use this inside a route by declaring the bean in Spring, say via the XML (or registering it in JNDI if that is your [Registry](http://camel.apache.org/registry.html))

|  |
| --- |
| <bean id="myProcessor" class="com.acme.MyProcessor"/> |

Then in Camel you can do

|  |
| --- |
| from("activemq:myQueue").to("myProcessor"); |

**Using the process DSL**

In your route you can also use the process DSL syntax for invoking a processor.

|  |
| --- |
| Processor myProcessor = new MyProcessor();  ...  from("activemq:myQueue").process(myProcessor); |

If you need to lookup the processor in the [Registry](http://camel.apache.org/registry.html) then you should use the **processRef** DSL:

|  |
| --- |
| from("activemq:myQueue").processRef("myProcessor"); |

**Why use process when you can use to instead?**

The process can be used in routes as an anonymous inner class such:

|  |
| --- |
| from("activemq:myQueue").process(new Processor() {      public void process(Exchange exchange) throws Exception {          String payload = exchange.getIn().getBody(String.class);          // do something with the payload and/or exchange here         exchange.getIn().setBody("Changed body");     }  }).to("activemq:myOtherQueue"); |

This is usable for quickly whirling up some code. If the code in the inner class gets a bit more complicated it is of course advised to refactor it into a separate class.

**Turning your processor into a full Component**

There is a base class called [ProcessorEndpoint](http://camel.apache.org/maven/current/camel-core/apidocs/org/apache/camel/impl/ProcessorEndpoint.html) which supports the full [Endpoint](http://camel.apache.org/endpoint.html) semantics given a Processor instance.

So you just need to create a [Component](http://camel.apache.org/component.html) class by deriving from [DefaultComponent](http://camel.apache.org/maven/current/camel-core/apidocs/org/apache/camel/impl/DefaultComponent.html) which returns instances of ProcessorEndpoint. For more details see [Writing Components](http://camel.apache.org/writing-components.html)

<http://camel.apache.org/processor.html>