JXPath

JXPath

Camel supports JXPath to allow XPath expressions to be used on beans in an Expression or Predicate to be used in the DSL or Xml Configuration. For example you could use JXPath to create an Predicate in a Message Filter or as an Expression for a Recipient List.

You can use XPath expressions directly using smart completion in your IDE as follows

from("queue:foo").filter(). jxpath("/in/body/foo"). to("queue:bar")

Variables

Variable	Type	Description
this	Exchange	the Exchange object
in	Message	the exchange.in message
out	Message	the exchange.out message

Options

Option	Type	Description
lenient	boolean	**Camel 2.11/2.10.5:** Allows to turn lenient on the JXPathContext. When turned on this allows the JXPath expression to evaluate against expressions and message bodies which may be invalid / missing data. See more details at the JXPath Documentation This option is by default false.

Using XML configuration

If you prefer to configure your routes in your Spring XML file then you can use JXPath expressions as follows

xml <beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation=" http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-spring.xsd"> <camelContext id="camel" xmlns="http://activemq.apache.org/camel/schema/spring"> <route> <from uri="activemq:MyQueue"/> <filter> <jxpath>in/body/name = 'James'</xpath> <to uri="mqseries:SomeOtherQueue"/> </filter> </route> < /camelContext> </beans>

Examples

Here is a simple example using a JXPath expression as a predicate in a Message Filter

{snippet:id=example|lang=java|url=camel/trunk/components/camel-jxpath/src/test/java/org/apache/camel/language/jxpath/JXPathFilterTest.java}

JXPath injection

You can use Bean Integration to invoke a method on a bean and use various languages such as JXPath to extract a value from the message and bind it to a method parameter.

For example

public class Foo { @MessageDriven(uri = "activemq:my.queue") public void doSomething(@JXPath("in/body/foo") String correlationID, @Body String body) { // process the inbound message here } }

Loading script from external resource

Available as of Camel 2.11

You can externalize the script and have Camel load it from a resource such as `"classpath:"`, `"file:"`, or `"http:"`.
This is done using the following syntax: `"resource:scheme:location"`, eg to refer to a file on the classpath you can do:

.setHeader("myHeader").jxpath("resource:classpath:myjxpath.txt")

Dependencies

To use JXpath in your camel routes you need to add the a dependency on **camel-jxpath** which implements the JXpath language.

If you use maven you could just add the following to your pom.xml, substituting the version number for the latest & greatest release (see the download page for the latest versions).

```
<dependency> <groupId>org.apache.camel</groupId> <artifactId>camel-jxpath</artifactId> <version>x.x.x</version> </dependency>
```

Otherwise, you'll also need Commons JXPath.