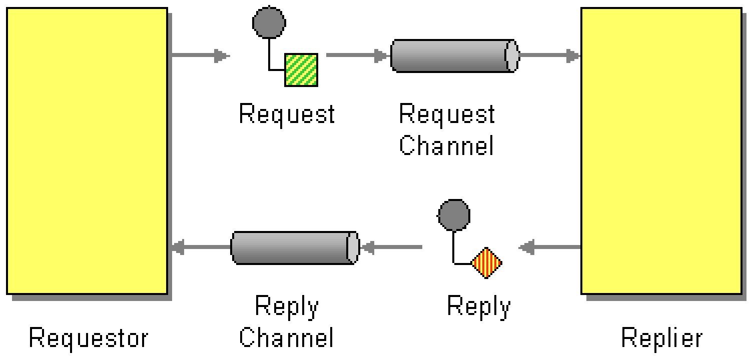
Request Reply

Request Reply

Camel supports the Request Reply from the EIP patterns by supporting the Exchange Pattern on a Message which can be set to **InOut** to indicate a request/reply. Camel Components then implement this pattern using the underlying transport or protocols.



For example when using JMS with InOut the component will by default perform these actions

- create by default a temporary inbound queue
- set the JMSReplyTo destination on the request message
- set the JMSCorrelationID on the request message
- send the request message
- consume the response and associate the inbound message to the request using the JMSCorrelationID (as you may be performing many concurrent request/responses).

Related

See the related Event Message message

Explicitly specifying InOut

When consuming messages from JMS a Request-Reply is indicated by the presence of the **JMSReplyTo** header.

You can explicitly force an endpoint to be in Request Reply mode by setting the exchange pattern on the URI. e.g.

```
jms:MyQueue?exchangePattern=InOut
```

You can specify the exchange pattern in DSL rule or Spring configuration.

Error formatting macro: snippet: java.lang.IndexOutOfBoundsException: Index: 20, Size: 20

Error rendering macro 'code' : Invalid value specified for parameter 'java.lang.NullPointerException'

```
<camelContext xmlns="http://camel.apache.org/schema/spring">
  <!-- Send the exchange as InOnly -->
  <route>
    <from uri="direct:testInOut"/>
    <inOut uri="mock:result"/>
  </route>

  <!-- Send the exchange as InOnly -->
  <route>
    <from uri="direct:testInOnly"/>
    <inOnly uri="mock:result"/>
  </route>


  <!-- lets set the exchange pattern then send it on -->
  <route>
    <from uri="direct:testSetToInOnlyThenTo"/>
    <setExchangePattern pattern="InOnly"/>
    <to uri="mock:result"/>
  </route>
  <route>
    <from uri="direct:testSetToInOutThenTo"/>
    <setExchangePattern pattern="InOut"/>
    <to uri="mock:result"/>
  </route>
  <route>
    <from uri="direct:testSetExchangePatternInOnly"/>
    <setExchangePattern pattern="InOnly"/>
    <to uri="mock:result"/>
  </route>


  <!-- Lets pass the pattern as an argument in the to element -->
  <route>
    <from uri="direct:testToWithInOnlyParam"/>
    <to uri="mock:result" pattern="InOnly"/>
  </route>
  <route>
    <from uri="direct:testToWithInOutParam"/>
    <to uri="mock:result" pattern="InOut"/>
  </route>
 </camelContext>
```

Using This Pattern

If you would like to use this EIP Pattern then please read the Getting Started, you may also find the Architecture useful particularly the description of Endpoint and URIs. Then you could try out some of the Examples first before trying this pattern out.