MDC logging

MDC logging

Available as of Camel 2.7

From **Camel 2.7**: Camel uses the SLF4J logging framework. This allows Camel to support MDC logging. For more information about about MDC logging see the logback manual.

The logging framework in use must support MDC. The following frameworks support MDC:

- log4j
- logback
- pax logging

See the logging framework's documentation for how to configure it to use MDC.

Enabling MDC Logging in Camel

To enable MDC logging using Java:

```
CamelContext context = ...
context.setUseMDCLogging(true);
...
```

To enable MDC logging using Spring XML set CamelContext's **`useMDCLogging`** attribute:

```
<camelContext xmlns="http://camel.apache.org/schema/spring" useMDCLogging="true">
  ...
</camelContext>
```

MDC Information

Camel provides the following context information available for MDC:

Before Camel 2.10:

Key	Description
`exchangeId`	The exchange id
`messageId`	**Camel 2.9.1:** The message id
`correlationId`	The correlation id of the exchange if it's correlated. For example a sub message from the Splitter EIP
`transactionKey`	The id of the transaction for transacted exchanges. Note the id is not unique, but its the id of the transaction template that marks the transaction boundary for the given transaction. Hence we decided to name the key `transactionKey` and not `transactionID` to point out this fact.
`routeId`	The id of the route, in which the exchange is currently being routed
`breadcrumbId`	**Camel 2.8:** An unique id used for tracking messages across transports.
`camelContextId`	**Camel 2.8.3/2.9:** the camel context id used for tracking the message from different camel context.

From Camel 2.10:

Key	Description

`camel.exchangeId`	The exchange id
`camel.messageId`	The message id
`camel.correlationId`	The correlation id of the exchange if it's correlated. For example a sub message from the Splitter EIP
`camel.transactionKey`	The id of the transaction for transacted exchanges. Note the id is not unique, but its the id of the transaction template that marks the transaction boundary for the given transaction. Hence we decided to name the key `transactionKey` and not `transactionID` to point out this fact.
`camel.routeId`	The id of the route, in which the exchange is currently being routed
`camel.breadcrumbId`	An unique id used for tracking messages across transports.
`camel.contextId`	The camel context id used for tracking the message from different camel context.

The keys are subject to change as we want to align and leverage MDC across other Apache products such as ActiveMQ, ServiceMix and Karaf.

Example Using LOG4J

If you use log4j you can configure MDC in the **`log4j.properties`** file as shown:

```
log4j.appender.out.layout.ConversionPattern=%d [%-15.15t] %-5p %-30.30c{1} - %-10.10X{camel.exchangeId} - %
-10.10X{camel.routeId} - %m%n
```

Camel will log on startup if MDC is enabled or not

```
INFO  SpringCamelContext -            -            - MDC logging is enabled on CamelContext: camel-1
```

The follow snippet is from an unit test which shows MDC in use. Note that the exchange id and route id are displayed in their separate columns in the log file:

```
INFO  SpringCamelContext -            -            - Apache Camel  (CamelContext: camel-1) started in 1.228
seconds
INFO  foo                - 358739-0-2 - route-a    - Exchange[ExchangePattern:InOnly, BodyType:String, Body:
Hello World]
INFO  bar                - 358739-0-2 - route-b    - Exchange[ExchangePattern:InOnly, BodyType:String, Body:
Hello World]
INFO  MockEndpoint       -            -            - Asserting: Endpoint[mock://result] is satisfied
```

Enabling Breadcrumb Support

From Camel 2.8:
The **`breadcrumbId`** key for MDC logging is only available if **`useBreadcrumb=true`** has been set on the CamelContext (default is true). When enabled Camel will enrich the Camel Message by adding a header to it with the key **`breadcrumbId`** containing the id. Camel will use the **`messageId`** if no existing **`breadcrumbId`** was found in the message.

As the breadcrumb id is stored in a header it will persist across any transport that supports the use of headers, for example the HTTP and JMS transports.

Disabling Breadcrumb Support

Java DSL:

```
CamelContext context = ...
context.setUseBreadcrumb(false);
...
```

XML DSL:

```
<camelContext xmlns="http://camel.apache.org/schema/spring" useBreadcrumb="false">
  ...
</camelContext>
```