StringTemplate

String Template

The **`string-template:`** component allows you to process a message using a String Template. This can be ideal when using Templating to generate responses for requests.

Maven users will need to add the following dependency to their **`pom.xml`** for this component:

xml<dependency> <groupId>org.apache.camel</groupId> <artifactId>camel-stringtemplate</artifactId> <version>x.x.x</version> <!-- use the same version as your Camel core version --> </dependency>

URI Format

string-template:templateName[?options]

Where **`templateName`** is the classpath-local URI of the template to invoke; or the complete URL of the remote template.

You can append query options to the URI in the following format, **`?option=value&option=value&...`**

Options

confluenceTableSmall

Option	Default	Description
`contentCache`	`false`	Cache for the resource content when its loaded. **Note**: as of **Camel 2.9** cached resource content can be cleared via JMX using the endpoint's **`clearContentCache`** operation.
`delimiterStart`	`null`	From **Camel 2.11.1**: configuring the variable start delimiter
`delimiterStop`	`null`	From **Camel 2.11.1**: configuring the variable end delimiter

Headers

Camel will store a reference to the resource in the message header with key, **`org.apache.camel.stringtemplate.resource`**. The Resource is an **`org.springframework.core.io.Resource`** object.

Hot-Reloading

The string template resource is by default hot-reloadable for both file and classpath resources (expanded jar). If you set **`contentCache=true`**, Camel will load the resource just once, disabling hot-reloading. This scenario can be used in production when the resource never changes.

StringTemplate Attributes

Camel will provide exchange information as attributes (just a **`java.util.Map`**) to the string template. The Exchange is transferred as:

confluenceTableSmall

Key	Value
`exchange`	The Exchange itself.
`headers`	The headers of the **`IN`** message.
`camelContext`	The Camel Context.
`request`	The **`IN`** message.
`in`	The **`IN`** message.
`body`	The **`IN`** message body.
`out`	The **`OUT`** message (only for **`InOut`** message exchange pattern).
`response`	The **`OUT`** message (only for **`InOut`** message exchange pattern).

From **Camel 2.14**: you can define the custom context map by setting the message header **`CamelStringTemplateVariableMap`**, as shown below:

javaMap<String, Object> variableMap = new HashMap<String, Object>(); Map<String, Object> headersMap = new HashMap<String, Object>(); headersMap.put("name", "Willem"); variableMap.put("headers", headersMap); variableMap.put("body", "Monday"); variableMap.put("exchange", exchange); exchange.getIn().setHeader("CamelStringTemplateVariableMap", variableMap);

Samples

For example you could use a string template as follows in order to formulate a response to a message:

from("activemq:My.Queue") .to("string-template:com/acme/MyResponse.tm");

The Email Sample

In this sample we want to use a string template to send an order confirmation email. The email template is laid out in **`StringTemplate`** as:
This example works for **camel 2.11.0**. If your camel version is less than **2.11.0**, the variables should be started and ended with **$**.

Dear <headers.lastName>, <headers.firstName> Thanks for the order of <headers.item>. Regards Camel Riders Bookstore <body>

And the java code is as follows:{snippet:id=e1|lang=java|url=camel/trunk/components/camel-stringtemplate/src/test/java/org/apache/camel/component/stringtemplate/StringTemplateLetterTest.java}Endpoint See Also