

# FreeMarker

## FreeMarker

The **freemarker** component allows for processing a message using a [FreeMarker](#) 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-freemarker</artifactId> <version>x.x.x</version> <!-- use the same version as your Camel core version --> </dependency>
```

### URI format

`freemarker:templateName[?options]`

Where **templateName** is the classpath-local URI of the template to invoke; or the complete URL of the remote template (eg: `file://folder/myfile.ftl`).

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

### Options

confluenceTableSmall

Option	Default	Description
contentCache	true	Cache for the resource content when it's loaded. Note: as of <b>Camel 2.9</b> cached resource content can be cleared via JMX using the endpoint's <code>clearContentCache</code> operation.
encoding	null	Character encoding of the resource content.
templateUpdateDelay	5	<b>Camel 2.9:</b> Number of seconds the loaded template resource will remain in the cache.

### Headers

Headers set during the FreeMarker evaluation are returned to the message and added as headers. This provides a mechanism for the FreeMarker component to return values to the Message.

An example: Set the header value of `fruit` in the FreeMarker template:

```
${request.setHeader('fruit', 'Apple')}
```

The header, `fruit`, is now accessible from the `message.out.headers`.

### FreeMarker Context

Camel will provide exchange information in the FreeMarker context (just a Map). The Exchange is transferred as:

confluenceTableSmall

key	value
exchange	The Exchange itself.
exchange.properties	The Exchange properties.
headers	The headers of the In message.
camelContext	The Camel Context.
request	The In message.
body	The In message body.
response	The Out message (only for InOut message exchange pattern).

From Camel 2.14, you can setup your custom FreeMarker context in the message header with the key "**CamelFreemarkerDataModel**" just like this

```
Map<String, Object> variableMap = new HashMap<String, Object>(); variableMap.put("headers", headersMap); variableMap.put("body", "Monday"); variableMap.put("exchange", exchange); exchange.getIn().setHeader("CamelFreemarkerDataModel", variableMap);
```

### Hot reloading

The FreeMarker template resource is by default **not** hot reloadable for both file and classpath resources (expanded jar). If you set `contentCache=false`, then Camel will not cache the resource and hot reloading is thus enabled. This scenario can be used in development.

## Dynamic templates

Camel provides two headers by which you can define a different resource location for a template or the template content itself. If any of these headers is set then Camel uses this over the endpoint configured resource. This allows you to provide a dynamic template at runtime.

confluenceTableSmall

Header	Type	Description	Support Version
FreemarkerConstants.FREEMARKER_RESOURCE	org.springframework.core.io.Resource	The template resource	<= 2.1
FreemarkerConstants.FREEMARKER_RESOURCE_URI	String	A URI for the template resource to use instead of the endpoint configured.	>= 2.1
FreemarkerConstants.FREEMARKER_TEMPLATE	String	The template to use instead of the endpoint configured.	>= 2.1

## Samples

For example you could use something like:

```
from("activemq:My.Queue"). to("freemarker:com/acme/MyResponse.ftl");
```

To use a FreeMarker template to formulate a response for a message for InOut message exchanges (where there is a `JMSReplyTo` header).

If you want to use InOnly and consume the message and send it to another destination you could use:

```
from("activemq:My.Queue"). to("freemarker:com/acme/MyResponse.ftl"). to("activemq:Another.Queue");
```

And to disable the content cache, e.g. for development usage where the `.ftl` template should be hot reloaded:

```
from("activemq:My.Queue"). to("freemarker:com/acme/MyResponse.ftl?contentCache=false"). to("activemq:Another.Queue");
```

And a file-based resource:

```
from("activemq:My.Queue"). to("freemarker:file://myfolder/MyResponse.ftl?contentCache=false"). to("activemq:Another.Queue");
```

In **Camel 2.1** it's possible to specify what template the component should use dynamically via a header, so for example:

```
from("direct:in"). setHeader(FreemarkerConstants.FREEMARKER_RESOURCE_URI).constant("path/to/my/template.ftl"). to("freemarker:dummy");
```

## The Email Sample

In this sample we want to use FreeMarker templating for an order confirmation email. The email template is laid out in FreeMarker as:

Dear \${headers.lastName}, \${headers.firstName} Thanks for the order of \${headers.item}. Regards Camel Riders Bookstore \${body}

And the java code:

```
{snippet:id=e1|lang=java|url=camel/trunk/components/camel-freemarker/src/test/java/org/apache/camel/component/freemarker/FreemarkerLetterTest.java}
```

[Endpoint See Also](#)