

# MVEL Component

## MVEL Component

Available as of Camel 2.12

The **mvel**: component allows you to process a message using an [MVEL](#) 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:

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

## URI format

```
mvel: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.mvel](#)).

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

## Options

| Option       | Default | Description   |
|--------------|---------|---|
| contentCache | true    | Cache for the resource content when it is loaded. The cached resource content can be cleared via JMX using the endpoint's <code>clearContentCache</code> operation. |
| encoding     | null    | Character encoding of the resource content.   |

## Message Headers

The `mvel` component sets a couple headers on the message.

| Header               | Description                                 |
|----------------------|---|
| CamelMvelResourceUri | The <b>templateName</b> as a String object. |

## MVEL Context

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

| key                 | value  |
|---------------------|--|
| exchange            | The <code>Exchange</code> itself.                          |
| exchange.properties | The <code>Exchange</code> properties.                      |
| headers             | The headers of the In message.                             |
| camelContext        | The Camel Context instance.                                |
| 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). |
|----------|--|

## Hot reloading

The mvel template resource is, by default, hot reloadable for both file and classpath resources (expanded jar). If you set `contentCache=true`, Camel will only load the resource once, and thus hot reloading is not possible. This scenario can be used in production, when the resource never changes.

## 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.

| Header               | Type   | Description  |
|----------------------|--------|--|
| CamelMvelResourceUri | String | A URI for the template resource to use instead of the endpoint configured. |
| CamelMvelTemplate    | String | The template to use instead of the endpoint configured.                    |

## Samples

For example you could use something like

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

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

To specify what template the component should use dynamically via a header, so for example:

```
from("direct:in").
  setHeader("CamelMvelResourceUri").constant("path/to/my/template.mvel").
  to("mvel:dummy");
```

To specify a template directly as a header the component should use dynamically via a header, so for example:

```
from("direct:in").
  setHeader("CamelMvelTemplate").constant("@{\\"The result is \\" + request.body * 3}\\\" }").
  to("velocity:dummy");
```

## See Also

- [Configuring Camel](#)
- [Component](#)
- [Endpoint](#)
- [Getting Started](#)