

Language

Language

Available as of Camel 2.5

The language component allows you to send [Exchange](#) to an endpoint which executes a script by any of the supported [Languages](#) in Camel. By having a component to execute language scripts, it allows more dynamic routing capabilities. For example by using the [Routing Slip](#) or [Dynamic Router](#) EIPs you can send messages to `language` endpoints where the script is dynamic defined as well.

This component is provided out of the box in `camel-core` and hence no additional JARs is needed. You only have to include additional Camel components if the language of choice mandates it, such as using [Groovy](#) or [JavaScript](#) languages.

URI format

```
language://languageName[:script][?options]
```

And from Camel 2.11 onwards you can refer to an external resource for the script using same notation as supported by the other [Languages](#) in Camel

```
language://languageName:resource:scheme:location[?options]
```

URI Options

The component supports the following options.

Name	Default Value	Type	Description
<code>languageName</code>	<code>null</code>	<code>String</code>	The name of the Language to use, such as <code>simple</code> , <code>groovy</code> , <code>javascript</code> etc. This option is mandatory.
<code>script</code>	<code>null</code>	<code>String</code>	The script to execute.
<code>transform</code>	<code>true</code>	<code>boolean</code>	Whether or not the result of the script should be used as the new message body. By setting to <code>false</code> the script is executed but the result of the script is discarded.
<code>contentCache</code>	<code>true</code>	<code>boolean</code>	Camel 2.9: Whether to cache the script if loaded from a resource. Note: from Camel 2.10.3 a cached script can be forced to reload at runtime via JMX using the <code>clearContentCache</code> operation.
<code>cacheScript</code>	<code>false</code>	<code>boolean</code>	Camel 2.13/2.12.2/2.11.3: Whether to cache the compiled script. Turning this option on can gain performance as the script is only compiled/created once, and reuse when processing Camel messages. But this may cause side-effects with data left from previous evaluation spills into the next, and concurrency issues as well. If the script being evaluated is idempotent then this option can be turned on.
<code>binary</code>	<code>false</code>	<code>boolean</code>	Camel 2.14.1: Whether the script is binary content. This is intended to be used for loading resources using the <code>Constant</code> language, such as loading binary files.

Message Headers

The following message headers can be used to affect the behavior of the component

Header	Description
<code>CamelLanguageScript</code>	The script to execute provided in the header. Takes precedence over script configured on the endpoint.

Examples

For example you can use the [Simple](#) language to [Message Translator](#) a message:

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

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

You can also use the [Groovy](#) language, such as this example where the input message will be multiplied with 2: `Error rendering macro 'code': Invalid value specified for parameter 'java.lang.NullPointerException'`

```
String script = URLEncoder.encode("request.body * 2", "UTF-8");  
from("direct:start").to("language:groovy:" + script).to("mock:result");
```

You can also provide the script as a header as shown below. Here we use [XPath](#) language to extract the text from the `<foo>` tag.

```
Object out = producer.requestBodyAndHeader("language:xpath", "<foo>Hello World</foo>", Exchange.  
LANGUAGE_SCRIPT, "/foo/text()");  
assertEquals("Hello World", out);
```

Loading scripts from resources

Available as of Camel 2.9

You can specify a resource uri for a script to load in either the endpoint uri, or in the `Exchange.LANGUAGE_SCRIPT` header. The uri must start with one of the following schemes: `file:`, `classpath:`, or `http:`

For example to load a script from the classpath:

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

By default the script is loaded once and cached. However you can disable the `concurrentCache` option and have the script loaded on each evaluation. For example if the file resource uri is `classpath:script.xml` then the endpoint definition would be:

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

and by prefixing with `resource:` as shown below:

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

See Also

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

- [Languages](#)
- [Routing Slip](#)
- [Dynamic Router](#)
- [Script](#)