**Simple Expression Language**

The Simple Expression Language was a really simple language when it was created, but has since grown more powerful. It is primarily intended for being a really small and simple language for evaluating [Expression](http://camel.apache.org/expression.html)s and [Predicate](http://camel.apache.org/predicate.html)s without requiring any new dependencies or knowledge of [XPath](http://camel.apache.org/xpath.html); so it is ideal for testing in **camel-core**. The idea was to cover 95% of the common use cases when you need a little bit of expression based script in your Camel routes.

However for much more complex use cases you are generally recommended to choose a more expressive and powerful language such as:

* [SpEL](http://camel.apache.org/spel.html)
* [Mvel](http://camel.apache.org/mvel.html)
* [Groovy](http://camel.apache.org/groovy.html)
* [JavaScript](http://camel.apache.org/javascript.html)
* [EL](http://camel.apache.org/el.html)
* [OGNL](http://camel.apache.org/ognl.html)
* one of the supported [Scripting Languages](http://camel.apache.org/scripting-languages.html)

The simple language uses **${body}** placeholders for complex expressions where the expression contains constant literals.

**Deprecated**: The **${}** placeholders can be omitted if the expression starts with the token, or if the token is only itself. (the token is like “body, header.foo”)

Alternative syntax

From Camel 2.5 you can also use the alternative syntax which uses **$simple{}** as placeholders. This can be used in situations to avoid clashes when using for example Spring property placeholder together with Camel.

Configuring result type

From Camel 2.8 you can configure the result type of the [Simple](http://camel.apache.org/simple.html) expression. For example to set the type as a **java.lang.Boolean** or a j**ava.lang.Integer** etc.

File language is now merged with Simple language

From Camel 2.2, the [File Language](http://camel.apache.org/file-language.html) is now merged with [Simple](http://camel.apache.org/simple.html) language which means you can use all the file syntax directly within the simple language.

Simple Language Changes in Camel 2.9 onwards

The [Simple](http://camel.apache.org/simple.html) language have been improved from Camel 2.9 to use a better syntax parser, which can do index precise error messages, so you know exactly what is wrong and where the problem is. For example if you have made a typo in one of the operators, then previously the parser would not be able to detect this, and cause the evaluation to be true. There are a few changes in the syntax which are no longer backwards compatible. When using [Simple](http://camel.apache.org/simple.html) language as a [Predicate](http://camel.apache.org/predicate.html) then the literal text **must** be enclosed in either single or double quotes. For example: "**${body} == 'Camel'**". Notice how we have single quotes around the literal. The old style of using "**body**" and "**header.foo**" to refer to the message body and header is **@deprecated**, and it is encouraged to always use **${}** tokens for the built-in functions.  
The range operator now requires the range to be in single quote as well as shown: "**${header.zip} between '30000..39999'**".

To get the body of the in message: **body**, or **in.body** or **${body}**.

A complex expression must use **${}** placeholders, such as: **Hello ${in.header.name} how are you?**.

You can have multiple functions in the same expression: "**Hello ${in.header.name} this is ${in.header.me} speaking**". However you can ***not*** nest functions in Camel 2.8.x or older e.g., having another **${}** placeholder in an existing, is not allowed. From **Camel 2.9** you can nest functions.

**Variables**

confluenceTableSmall

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| camelId | String | **Camel 2.10:** the [CamelContext](http://camel.apache.org/camelcontext.html) name. |
| camelContext.**OGNL** | Object | **Camel 2.11:** the **CamelContext** invoked using a Camel OGNL expression. |
| collate(*group*) | List | **Camel 2.17:** The collate function iterates the message body and groups the data into sub lists of specified size. This can be used with the [Splitter](http://camel.apache.org/splitter.html) EIP to split a message body and group/batch the split sub messages into a group of **N** sub lists. This method works similar to the collate method in Groovy. |
| exchange | Exchange | **Camel 2.16:** the [Exchange.](http://camel.apache.org/exchange.html) |
| exchange.**OGNL** | Object | **Camel 2.16:** the [Exchange](http://camel.apache.org/exchange.html) invoked using a Camel OGNL expression. |
| exchangeId | String | **Camel 2.3:** the exchange Id. |
| id | String | The input message Id. |
| body | Object | The input body. |
| header.foo | Object | Refer to the input foo header. |

<http://camel.apache.org/simple.html>