

Schematron

Schematron Component

Available as of Camel 2.14

Schematron is an XML-based language for validating XML instance documents. It is used to make assertions about data in an XML document and it is also used to express operational and business rules. Schematron is an [ISO Standard](#). The schematron component uses the leading [implementation](#) of ISO schematron. It is an XSLT based implementation. The schematron rules is run through [four XSLT pipelines](#), which generates a final XSLT which will be used as the basis for running the assertion against the XML document. The component is written in a way that Schematron rules are loaded at the start of the endpoint (only once) this is to minimise the overhead of instantiating a Java Templates object representing the rules.

URI format

```
schematron://path?[options]
```

URI options

Name	Default value	Description
path	mandatory	The path to the schematron rules file. Can either be in class path or location in the file system.
abort	false	flag to abort the route and throw a schematron validation exception.

Headers

Name	Description	Type	In/Out
CamelSchematronValidationStatus	The schematron validation status: SUCCESS / FAILED	String	IN
CamelSchematronValidationReport	The schematron report body in XML format. See an example below	String	IN

URI and path syntax

The following example shows how to invoke the schematron processor in Java DSL. The schematron rules file is sourced from the class path:

```
from("direct:start").to("schematron://sch/schematron.sch").to("mock:result")
```

The following example shows how to invoke the schematron processor in XML DSL. The schematron rules file is sourced from the file system:

```
<route>
  <from uri="direct:start" />
  <to uri="schematron:///usr/local/sch/schematron.sch" />
  <log message="Schematron validation status: ${in.header.CamelSchematronValidationStatus}" />
  <choice>
    <when>
      <simple>${in.header.CamelSchematronValidationStatus} == 'SUCCESS'</simple>
      <to uri="mock:success" />
    </when>
    <otherwise>
      <log message="Failed schematron validation" />
      <setBody>
        <header>CamelSchematronValidationReport</header>
      </setBody>
      <to uri="mock:failure" />
    </otherwise>
  </choice>
</route>
```

Where to store schematron rules?

Schematron rules can change with business requirement, as such it is recommended to store these rules somewhere in file system. When the schematron component endpoint is started, the rules are compiled into XSLT as a Java Templates Object. This is done only once to minimise the overhead of instantiating Java Templates object, which can be an expensive operation for large set of rules and given that the process goes through four pipelines of [XSLT transformations](#). So if you happen to store the rules in the file system, in the event of an update, all you need is to restart the route or the component. No harm in storing these rules in the class path though, but you will have to build and deploy the component to pick up the changes.

Schematron rules and report samples

Here is an example of schematron rules

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
  <title>Check Sections 12/07</title>
  <pattern id="section-check">
    <rule context="section">
      <assert test="title">This section has no title</assert>
      <assert test="para">This section has no paragraphs</assert>
    </rule>
  </pattern>
</schema>
```

Here is an example of schematron report:

```
<?xml version="1.0" encoding="UTF-8"?>
<svrl:schematron-output xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
  xmlns:iso="http://purl.oclc.org/dsdl/schematron"
  xmlns:saxon="http://saxon.sf.net/"
  xmlns:schold="http://www.ascc.net/xml/schematron"
  xmlns:xhtml="http://www.w3.org/1999/xhtml"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" schemaVersion="" title="">

  <svrl:active-pattern document="" />
  <svrl:fired-rule context="chapter" />
  <svrl:failed-assert test="title" location="/doc[1]/chapter[1]">
    <svrl:text>A chapter should have a title</svrl:text>
  </svrl:failed-assert>
  <svrl:fired-rule context="chapter" />
  <svrl:failed-assert test="title" location="/doc[1]/chapter[2]">
    <svrl:text>A chapter should have a title</svrl:text>
  </svrl:failed-assert>
  <svrl:fired-rule context="chapter" />
</svrl:schematron-output>
```

Useful Links and resources



- [Introduction to Schematron](#) by Mulleberry technologies. An excellent document in PDF to get you started on Schematron.
- [Schematron official site](#). This contains links to other resources