Using Schematron for Ant by example

The schematron-ant.jar task allows ISO Schematron validation and reporting from an ANT script. The software is available open source from http://www.schematron.com/ and was contributed by companies Allette Systems, WebOrganic and Topologi.

When used with SAXON 9 XSLT, schematron-ant.jar will allow schemas with ISO Schematron (XSLT2 and XSLT2) query language bindings, and also obsolete Schematron 1.6 schemas. The output can be ISO SVRL (Schematron Validation Report Language, an XML format) or plain messages.

Installing the Ant task

Before the schematron Ant Task can be setup the following must be installed on the system:
- Any recent Java (JRE or JDK; 1.4.2 or later)
- Any recent Apache Ant (1.6.0 or later)
- Any JAXP-conforming library for XSLT which includes DOMSource and DOMResult implementations)
  - We test using Saxon 9 Home Edition - saxon9he.jar
  - Other versions of SAXON 9 may work, but as well as saxon9.jar you need saxon-dom.jar

Details on how to configure Ant to run on a system can be found on:
> http://ant.apache.org/manual/

In order to use Schematron Ant task, you simply need to have the 'ant-schematron.jar' file.

It can either use for a specific project and placed in the project folder.

If you would like to install the schematron Ant library for you system, you can put it in the Ant library folder (generally C:\Program Files\Ant\lib), so that Ant can detect it automatically.

Defining the Ant task

To use the Ant task, it must be defined in your build file (generally build.xml).

This can be done using the <taskdef> Ant element in your Ant file.

```
<taskdef name="schematron"
      classname="com.schematron.ant.SchematronTask"
      classpath="lib/ant-schematron.jar"/>
```

Or if the Jar file is available in your system, simply.

```
<taskdef name="schematron" classname="com.schematron.ant.SchematronTask"/>
```

The task can be defined at the top level or within a specific target.

Using the Ant task

Once the Schematron Ant Task has been defined, you can use the <schematron> element to tell Ant to validate a bunch of files with Schematron. Both ISO Schematron and the obsolete Schematron 1.n are supported.

Parameters

Common parameters:

Attribute	Value	Description	Required
schema		the path to the schematron schema file.	Yes
file		the file(s) you want to check. (optionally can use an embedded fileset)	Only if no fileset is defined
outputFilename		The output file name of the SVRL file. The default is "result.xml"	No
OutputDir		The output directory of the SVRL file. The default is the current working directory of the ANT task.	No
phase		the ISO Schematron phase to use (ignored if no phase was specified in the schema)	No
format	¡°svrl" (default)| "message" | ¡°erminate"	SVRL is an XML format for the results. "message" is a simple text format. ¡°terminate" is a text format and halts at first error. (Not implemented yet.)	No
queryLanguageBinding	¡°xslt1" (default)| "xslt2" | "xpath" | "xpath2" |"old"	Select to match the schema. ¡°old" is for Schematron 1.n, e.g. Schematron 1.5	No
allow_foreign	"true" | "false" (default)	Pass non-Schematron elements to the generated stylesheet. Pass the Schematron elements span, emph and dir: to the output SVRL.	No
failOnError	"true" | "false" (default)	Return a fail error code to Ant on an assertion failure. This also enables file logging.	No
resolver	classname	The classname for a URL resolver. Must implement javax.xml.transform.URIResolver	No
classpath	string	The path to the class or jar for that resolver.	No
xml.catalog.files	string	The semicolon-delimited list of catalog files, if the resolver expects this. (Passed as system parameter.)	No
fileNameParameter	string	A parameter passed to the Validator and potentially available as a variable in Schematron schemas as $fileNameParameter	No
fileDirParameter	string	A parameter passed to the Validator and potentially available as a variable in Schematron schemas as $fileDirParameter	No
archiveNameParameter	string	A parameter passed to the Validator and potentially available as a variable in Schematron schemas as $archiveNameParameter	No
archiveDirParameter	string	A parameter passed to the Validator and potentially available as a variable in Schematron schemas as $archivePathParameter	No
debugMode	¡°true" | "false" (default)	Emit the XSLT generated from the Schematron schemas as the file debug.xslt	No

Parameters not used by every formatter

Attribute	Value	Description	Required

generate-paths	true|false	generate the SVRL @location attribute with XPaths	No
diagnose	yes | no	Add the diagnostics to the assertion results	No
terminate	yes | no	Terminate on the first failed assertion or successful report	No
message-newline	"true" (default) | "false"	Generate an extra newline at the end of messages	No
output-encoding	string	The encoding used for the SVRL output	No

Legacy, rare, experimental or compatibility parameters

Attribute	Value	Description	Required
sch.exslt.imports	semi-colon delimited string of filenames	Used for some EXSLT implementations (Not tested with Schematron Ant)	No
attributes	"true" | "false" (Autodetecting)	Use only when the schema has no attributes as the context nodes	No
only-child-elements	"true" | "false" (Autodetecting)	Use only when the schema has no comments or PI as the context nodes	No
visit-text	"true" | "false" (default)	Also visit text nodes for context.	No
select-contents	'' | 'key' | '//'	Select different implementation strategies	No
property	¡°true" | "false"	Generate property	No

Typically, you would use a `<fileset>` to specify the list of files that should be validated. Filesets are a standard Ant concept, they are used to specify lists of files matching certain criteria based on the name or path. More details can be found on:

> http://ant.apache.org/manual/CoreTypes/fileset.html

The Ant task will produce results both:
- On the console, in a concise and informative way for the user to know what is going on
- And as an SVRL file which contains more verbose information in XML that can be used for better diagnostics and possibly further processing.

Running Ant

On the DOS console, simply execute ant in the directory followed by the name of the build file
- ant [build.xml]

Within Eclipse, simply right-click on the ant file and select 'run'.

Ant task example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!--
   A simple Ant script to illustrate how to use the Schematron Ant Task.

  This Ant files uses the 'ant-schematron.jar' file which contains the implementation
for the Ant task as well as the ISO Schematron pre-processor (XSLT).

  @author Christophe Lauret
   @author Willy Ekasalim

  @version 16 February 2007
```

```
-->
<project name="schematron-ant-sample" default="validate">

  <!--
     This is a task definition for Ant, it allows Ant to map the 'schematron' element
name to the schematron task that it loads from the Jar file.
  -->
  <taskdef name="schematron"
          classname="com.schematron.ant.SchematronTask"
          classpath="lib/ant-schematron.jar"/>

  <!--
     Validate the files specified in the fileset using the 'sample.sch' schematorn file.
       Concise results will be displayed on the console for each file.
       More verbose results are generated as SVRL format and saved as 'result.xml'
  -->
  <target name="validate" description="Test with a Fileset">
      <schematron schema="sch/sample.sch" failonerror="false">
          <fileset dir="xml" includes="*.xml"/>
      </schematron>
  </target>

</project>
```

Sample Source XML

Sample1.xml	*Sample2.xml*	*Sample3.xml*

`<?xml version="1.0"?>`
`<Dog>`
`  <leg>`
`    <paw></paw>`
`  </leg>`
`  <leg>`
`    <paw></paw>`
`  </leg>`
`  <leg>`
`    <paw></paw>`
`  </leg>`
`  <leg>`
`    <paw></paw>`
`  </leg>`
`</Dog>`
	`<?xml version="1.0"?>`
	`<Dog>`
	`  <leg>`
	`     <paw></paw>`
	`  </leg>`
	`  <leg>`
	`    <paw></paw>`
	`  </leg>`
	`  <leg>`
	`  </leg>`
	`</Dog>`
		`<?xml version="1.0"?>`
		`<Dog>`
		`  <leg>`
		`     <paw></paw>`
		`  </leg>`
		`  <leg>`
		`     <paw></paw>`
		`  </leg>`
		`</Dog>`

Sample Schematron Schema

```
<!--
   Sample schema for use with the Schematron Ant task.

   @version 16 February 2007
-->
<schema xmlns="http://purl.oclc.org/dsdl/schematron">

   <title>Dog Stuff</title>

   <pattern>
       <rule context="Dog">
       <assert test="count(leg) = 4">A dog should have four legs, because then they can
have four paws.</assert>
             <report test="count(leg) &lt; 3">A dog with less than three legs is
unstable</report>
       </rule>
       <rule context="Dog/leg">
```

```
                <assert test="count(paw) = 1">Each dog's leg should have a single paw, as
an element or attribute, because this meets the business requirement "Dog must be
walkable".</assert>
        </rule>
    </pattern>
</schema>
```

Console Results

This is what Ant will produce on the system console:

```
validate:
[schematron] Source file: sample2.xml
[schematron] [assert] /Dog[1] - A dog should have four legs, because then they can have
four paws.
[schematron] [assert] /Dog[1]/leg[3] - Each dog's leg should have a single paw, as an
element or attribute, because this meets the business requirement "Dog must be walkable".
[schematron] Source file: sample3.xml
[schematron] [assert] /Dog[1] - A dog should have four legs, because then they can have
four paws.
[schematron] [report] /Dog[1] - A dog with less than three legs is unstable
[schematron] 3 file(s) have been successfully validated.
BUILD SUCCESSFUL
Total time: 3 seconds
```

SVRL Results

This is the XML that Ant will produce and save as 'result.xml':

```
<fileset date="2007/02/16">

  <file name="sample2.xml">
      <svrl:schematron-output xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
xmlns:sch="http://www.ascc.net/xml/schematron"
xmlns:iso="http://purl.oclc.org/dsdl/schematron"
        xmlns:xs="http://www.w3.org/2001/XMLSchema" title="Dog Stuff" schemaVersion="">
        <svrl:active-pattern/>
        <svrl:fired-rule context="Dog"/>
        <svrl:failed-assert test="count(leg) = 4" location="/Dog[1]">
            <svrl:text>A dog should have four legs, because then they can have four
paws.</svrl:text>
        </svrl:failed-assert>
        <svrl:fired-rule context="Dog/leg"/>
        <svrl:fired-rule context="Dog/leg"/>
        <svrl:fired-rule context="Dog/leg"/>
        <svrl:failed-assert test="count(paw) = 1" location="/Dog[1]/leg[3]">
            <svrl:text>Each dog's leg should have a single paw, as an element or
attribute, because this meets the business requirement "Dog must be walkable".</svrl:text>
        </svrl:failed-assert>
      </svrl:schematron-output>
  </file>




  <file name="sample1.xml">
      <svrl:schematron-output xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
xmlns:sch="http://www.ascc.net/xml/schematron"
xmlns:iso="http://purl.oclc.org/dsdl/schematron"
        xmlns:xs="http://www.w3.org/2001/XMLSchema" title="Dog Stuff" schemaVersion="">
        <svrl:active-pattern/>
        <svrl:fired-rule context="Dog"/>
        <svrl:fired-rule context="Dog/leg"/>
        <svrl:fired-rule context="Dog/leg"/>
        <svrl:fired-rule context="Dog/leg"/>
        <svrl:fired-rule context="Dog/leg"/>
      </svrl:schematron-output>
  </file>
```

```xml
  <file name="sample3.xml">
      <svrl:schematron-output xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
xmlns:sch="http://www.ascc.net/xml/schematron"
xmlns:iso="http://purl.oclc.org/dsdl/schematron"
          xmlns:xs="http://www.w3.org/2001/XMLSchema" title="Dog Stuff" schemaVersion="">
          <svrl:active-pattern/>
          <svrl:fired-rule context="Dog"/>
          <svrl:failed-assert test="count(leg) = 4" location="/Dog[1]">
              <svrl:text>A dog should have four legs, because then they can have four
paws.</svrl:text>
          </svrl:failed-assert>
          <svrl:successful-report test="count(leg) &lt; 3" location="/Dog[1]">
              <svrl:text>A dog with less than three legs is unstable</svrl:text>
          </svrl:successful-report>
          <svrl:fired-rule context="Dog/leg"/>
          <svrl:fired-rule context="Dog/leg"/>
      </svrl:schematron-output>
  </file>

</fileset>
```