

ValidatingTransformer

The `ValidatingTransformer` provides a very simple `Transformer` implementation validating documents in a pipeline using the validation framework provided by the [validation block](#)¹.

The configuration for this transformer is extremely easy: Like any other component, it needs to be declared in a sitemap's `<map:components/>` section:

```
<map:components>
  ...
  <map:transformers default="...">
    ...
    <map:transformer name="validate"
      logger="sitemap.transformer.validate"
      src="org.apache.cocoon.transformation.ValidatingTransformer">

      <!--+ The "grammar" an optional configuration element specifying the default
          grammar used by the validator. When the "grammar" element is specified,
          the automatic detection of schema types will be disabled.

          This can also be specified in pipelines as a parameter:
          <map:parameter name="grammar" value="... grammar identifier ..."/>

          For a list of well known grammars, see the JavaDoc documentation for the
          org.apache.cocoon.components.validation.Validator interface.
      +-->

      <!-- <grammar>... grammar identifier ...</grammar> -->
    </map:transformer>

    ...
  </map:transformers>
</map:components>
```

The only defined (but **not required**) configuration element for this component is `<grammar>... grammar identifier ...</grammar>` indicating the grammar to pass to the `Validator` component. See the documentation of the [validation block](#)² for more information on what this means.

The use of the `ValidatingTransformer` is again extremely simple. Simply declare it in the pipeline like any other transformer:

```
<map:pipeline type="...">
  ...
  <map:match src="...">
    <map:generate src="...">
      ...
      <map:transform type="validating" src="myschema.rng">
        <!-- <map:parameter name="grammar" value="... identifier ..."/> -->
      </map:transform>
      ...
    <map:serialize type="...">
  </map:match>

  ...
</map:pipeline>
```

Following the example above, the document being processed in the pipeline will be validated using the `"myschema.rng"` schema (the `Validator` will autodetect this to be a [RELAX-NG](#)³ schema), and the entire pipeline will fail throwing a `SAXException` if an error was detected validating.

-
1. [daisy:683 \(Validation\)](#)
 2. [daisy:683#grammars \(Validation\)](#)
 3. <http://relax-ng.org/>

The optional "grammar" parameter will override at run time whatever configuration parameter was specified in the `<map:components/>` section of the pipeline. For example one might not declare a grammar in the configuration (letting the normal grammar auto-detection to work) and override it for a specific pipeline with the XML DTD grammar identifier (which can't be detected, as it's not an XML document).

Note that if multiple errors occur in any document, only the first one will be reported: this transformer doesn't give you a report of all errors in the document, it simply fail on the first error detected. Another transformer, the [ValidationReportTransformer](#)⁴ allows to produce validation reports.

Fields

Name	Value
FQ Java class	org.apache.cocoon.transformation.ValidatingTransformer
CocoonComponentReference	Transformer
CocoonBlock	validation
Cacheability info (DO NOT EDIT)	Yes

4. daisy:691 (ValidationReportTransformer)