parsQube

Greenfox

A schema language for validating file systems

Hans-Jürgen Rennau, parsQube GmbH
Presented at xmlprague 2020, February 15, 2020

Greenfox: definition

Greenfox is a language for validating file system trees* against a set of conditions.

**file system tree =*
a folder + all folders and files directly or indirectly contained

File system validation

- Why?
- What, precisely?
- How?

Conventional validation – some basic limitations

Limitation 1:
The **scope is limited** to a set of related document types, belonging to a single mediatype

Limitation 2:
Files are the input – their presence or absence is out of scope, whereas the real problem may be the **absence of a file**

Limitation 3:
Expectations about resources are static – **ignoring dependence** on other resources (their presence and contents)

What, precisely?

- Folder contents
- File contents
  - Schema-valid (XSD, JSON Schema, SHACL, …)
  - Rules conformant („*If contains Foo, contains Bar*“)
- Folder/file content dependencies:
  - File A exists <=> file B exists
  - File A exists <=> file B contains `<Bar>`
  - File A contains `<Foo>` <=> file B contains `<Bar>`

A first schema, with a folder shape

```
<greenfox greenfoxURI="http://www.greenfox.org/ns/schema-examples/example-folder-content"
          xmlns="http://www.greenfox.org/ns/schema">
    <!-- File system tree = Oxygen installation -->
    <domain path="[redacted]" name="oxygen19">

        <!-- *** Oxygen folder "xmlschema" *** -->
        <folder foxpath="[redacted] id="xmlschemaFolderShape">
            <targetSize msg="No xmlschema folder found" minCount="1"/>
            <folderContent closed="true" ignoredMembers="*.html"
                           closedMsg="xmlschema folder with unexpected content.">
                <memberFolders names="dtd, templates, xsd"/>
                <memberFile name="*.css"
                            minCount="5" minCountMsg="Not enough css files"
                            maxCount="7" maxCountMsg="Too many css files"/>
                <memberFile name="catalog.xml" minCountMsg="Missing catalog file"/>
                <memberFile name="xmlschema.jar" minCountMsg="Missing jar file"
                            md5="BD509F9C80642BC03856B1DF169D0EA3"
                            md5Msg="Not expected MD5"/>
                <memberFile name="xmlschema.framework"/>
            </folderContent>
        </folder>
    </domain>
</greenfox>
```

A folder shape

```
<folder [redacted] id="xmlschemaFolderShape">
    <targetSize msg="No xmlschema folder found" minCount="1"/>
    <folderContent closed="true" ignoredMembers="*.html"
                   closedMsg="xmlschema folder with unexpected content.">
        <memberFolders names="dtd, templates, xsd"/>
        <memberFile name="*.css"
                    minCount="5" minCountMsg="Not enough css files"
                    maxCount="7" maxCountMsg="Too many css files"/>
        <memberFile name="catalog.xml" minCountMsg="Missing catalog file"/>
        <memberFile name="xmlschema.jar" minCountMsg="Missing jar file"
                    md5="BD509F9C80642BC03856B1DF169D0EA3"
                    md5Msg="Not expected MD5"/>
        <memberFile name="xmlschema.framework"/>
    </folderContent>
</folder>
```

Features to remember

(a) wildcards + min/maxCount (b) hash keys (c) closed?

A file shape

```
<!-- *** Catalog file *** -->
<file [redacted] id="catalogFileShape">
    <targetSize countMsg="Catalog file not found" count="1"/>
    <lastModified lt="2020-01-01"
                  ltMsg="Must be older than 2020-01-01"/>
    <xpath expr="//*/namespace-uri(.)"
           eq="urn:oasis:names:tc:entity:xmlns:xml:catalog"
           eqMsg="Unexpected namespace."/>
    <xpath expr="//@uri"
           like="dtd/*.dtd|xsd/*.xsd"
           likeMsg="Unexpected URI"/>
</file>
```

A folder shape, containing a file shape

```
<folder foxpath="[redacted]" id="xmlschemaFolderShape">
    <targetSize msg="No xmlschema folder found" minCount="1"/>
    <folderContent closed="true" ignoredMembers="*.html" [9 lines]

    <!-- *** Catalog file *** -->
    [redacted]

</folder>
```

Nesting shapes - subsetting

```
<!-- docbook files -->
<file foxpath="[redacted]">
    ...
    <!-- docbook with images -->
    <file foxpath="[redacted]">
      ...
```

Nesting shapes – if outer target, then inner

```
<!-- docbook files -->
<file foxpath="[redacted]">
    ...
    <!-- docbook with images -->
    <file foxpath="[redacted]">
      ...
      <folder foxpath="[redacted]">
        <targetSize countMsg="No img subfolder"
                    count="1"/>
      </folder>
    </file>
</file>
```

Validating JSON contents

```
<!-- File system tree = Oxygen installation -->
<domain path="\programme\Oxygen XML Editor 19" name="oxygen19">

    <!-- *** dita package.json *** -->
    <file foxpath=".\\reveal.js-2.6.2\package.json" id="jsonFileShape"
          >
        <targetSize msg="Dita package json not found" count="1"/>
        <xpath expr="//version"
               count="1"
               countMsg="Not one version found"
               eq="2.6.2"
               eqMsg="Not the expected version (2.6.2)"/>
        <xpath expr="//*/local-name(.)"
               notMatches="_[^_]"
               notMatchesMsg="JSON name not an NCName"/>
    </file>
</domain>
```

Key feature #1 - XDM based

Value = XDM value

- Value = **sequence of items**
- Item
  - Atomic value (with a type from xsd)
  - XDM node (element, attribute, …)
  - Map or array (rarely used)
  - *Function item* *(currently not used)*

Key feature #2

Navigation skills

- **Between** resources — **foxpath**
- **Within** resources — **XPath**
- **Multi-mediatype** — **XPath - extensions**
- **Cross-boundary**: — **foxpath**
  - Start inside file, …… || …… arrive at a file-or-folder
  - Start at a file-or-folder, …… || …… arrive inside file
  - Start inside file A, ……..... || …… arrive inside file B

Key feature #3

Validation concept

Shapes Constraint Language (SHACL)

W3C Recommendation 20 July 2017

This version:
https://www.w3.org/TR/2017/REC-shacl-20170720/

Latest published version:
https://www.w3.org/TR/shacl/

Latest editor's draft:
https://w3c.github.io/data-shapes/shacl/

Implementation report:
https://w3c.github.io/data-shapes/data-shapes-test-suite/

Previous version:
https://www.w3.org/TR/2017/PR-shacl-20170608/

Editors:
Holger Knublauch, TopQuadrant, Inc.
Dimitris Kontokostas, University of Leipzig

Repository:
GitHub
Issues

Test Suite:
SHACL Test Suite

Concepts

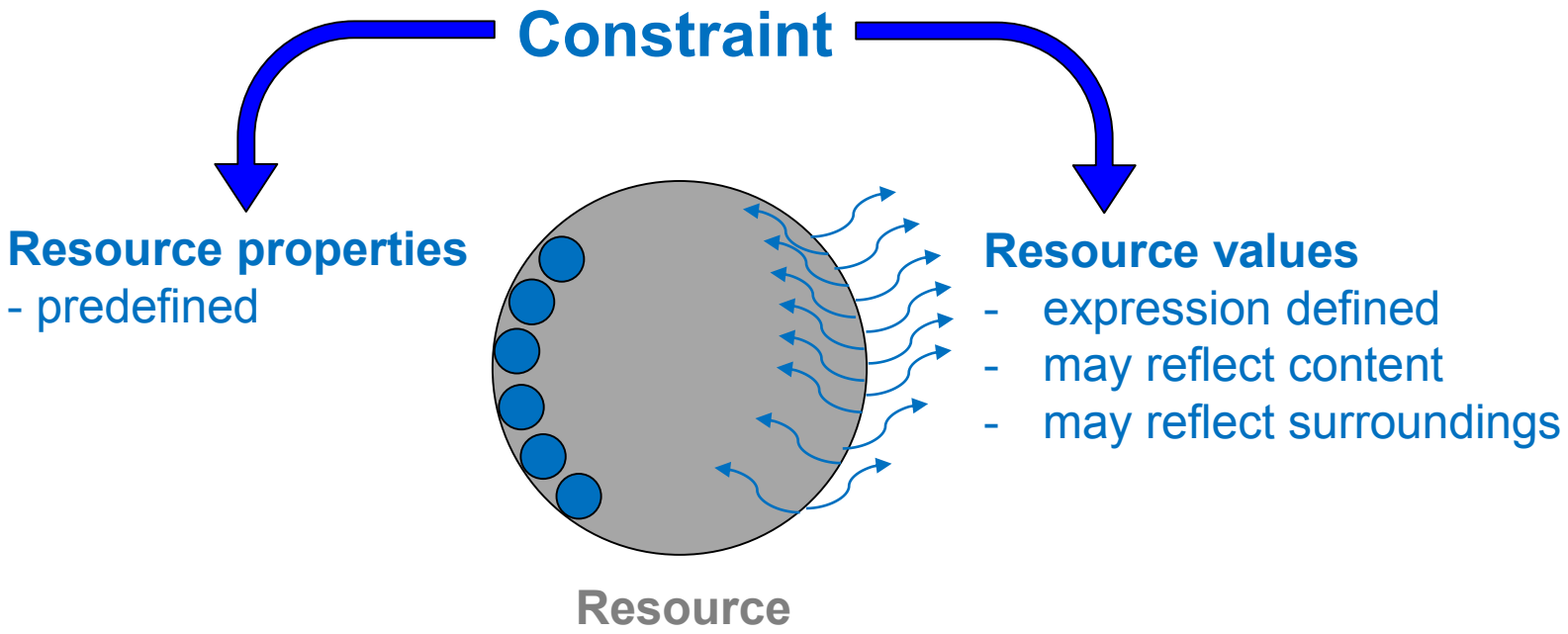
- Resource — file, folder
- Shape — a set of constraints
- Constraint — a condition to be checked

- Resource shape — checks a **resource**
  - Target declaration — *selects resources*
  - Constraints — *check <u>resource properties</u>*
- Value shape — checks a **resource value**
  - Expression — *constructs a resource value*
  - Constraints — *check the <u>resource value</u>*

Constraints

- A constraint is *declared* by a shape
- Constraint declaration: like a function call
  - „function name“ = Constraint Component name
  - „call parameters“ = Constraint Parameter values
- Validation:
  - Input:
    - parameter values
    - resource or resource value
  - Output: validation result

What a constraint perceives



Constraining resource values

Value shape, declaring a constraint

```
<xpath
    expr="//airport[@href and *]"
    empty="true"
    emptyMsg="Airport elements should have EITHER @href OR content"/>
```

Validation result

```
<gx:red filePath="C:/tt/greenfox/example-data/airports/airports.xml"
  msg="Airport elements should have EITHER @href OR content"
  constraintComp="ExprValueEmpty"
  constraintID="hrefOrChildren-empty"
  valueShapeID="hrefOrChildren"
  valueCount="2"
  exprLang="xpath"
  expr="//airport[@href and *]">
  <gx:valueNodePath>/airports/airport[1]</gx:valueNodePath>
  <gx:valueNodePath>/airports/airport[4]</gx:valueNodePath>
</gx:red>
```

Expression tandems

```
<file foxpath="projectDates.xml">
  <xpath
        expr="[redacted]"
        gtXPath="[redacted]"
        gtXPathContext="#item"
        gtXPathMsg="milestone date before project start"/>
</file>
```

Example: folder resource values

```
<!-- *** Any folder in the domain *** -->
<folder foxpath=".\\*[is-dir(.)]" id="goodFolderShape">

    <!-- Resource value empty files
    <foxpath expr="*[is-file()][file-size(.) eq 0]"
            empty="true" emptyMsg="Empty files"/>
    <!-- Resource value ill-formed XML
    <foxpath expr="*.xml[not(is-xml(.))]"
            empty="true" emptyMsg="Invalid XML files"/>
    <!-- Resource value ill-formed JSON
    <foxpath expr="*.json[not(is-json(.))]"
            empty="true" emptyMsg="Invalid JSON files"/>
</folder>
```

Good folder shape

Example:
external resource value

```
<!-- *** Response file shape *** -->
<file foxpath=".\\test-*[config]\output\(*.xml, *.json)"
      mediatype="xml-or-json">

    <!-- # Check - return code ok? -->
    <foxpath
        expr="..\..\config\msg-config.csv\csv-doc(., ',', 'yes')
              //record[response eq $fileName]/returnCode"
        eqXPath="//*:returnCode"
        eqXPathMsg="Return code not the configured value"/>

</file>
```

Resource value *expected return code*, fetched from a distant CSV

Exploring files with shifting focus nodes

```
<[redacted] foxpath="factbook.xml" id="factbookFileShape">
    <targetSize countMsg="No factbook file found" count="1"/>
    <[redacted] xpath="/mondial/country">
        <xpath expr="@capital" eqXPath=".//city/@id"
               eqXPathMsg="Country capital must reference a city element contained."/>
        <[redacted] xpath="province">
            <xpath expr="@capital" eqXPath="city/@id"
                   eqXPathMsg="Province capital must reference a city element contained."/>
            <[redacted] xpath="city">
                <xpath expr="@country" eqXPath="ancestor::country/@id"
                       eqXPathMsg="City must reference containing country"/>
                <xpath expr="name" minCount="1"
                       minCountMsg="City must have at least one name child"/>
            </focusNode>
        </focusNode>
    </focusNode>
</file>
```

Constraint components

Resource shapes

TargetSize*, LastUpdate*, FileSize*, Mediatype*, FolderContent*, xsdValid

Value shapes I: Literal parameters

itemsUnique
empty exists
count minCount maxCount
datatype
eq ne gt ge lt le
length minLength maxLength
matches notMatches
like notLike

Value shapes II: Expression valued parameters

eqFoxpath neFoxpath ltFoxpath leFoxpath gtFoxpath geFoxpath
eqXPath neXPath ltXPath leXPath gtXPath geXPath
inFoxpath containsFoxpath
inXPath containsXPath

User-defined constraints

Constraint definition

```
<!-- Simple grep constraint -->
<constraintComponent constraintElementName="[redacted]">
    <param name="pattern" type="xs:string"></param>
    <param name="flags" type="xs:string?"></param>
    <validatorXPath>
    exists(unparsed-text-lines($this)[matches(., $pattern, $flags)])
    </validatorXPath>
</constraintComponent>
```

Constraint declaration

```
<file foxpath="README.md">
    <targetSize msg="README file not found" minCount="1"/>
    <[redacted] pattern="ISO 639-3" flags="i"
            msg="File does not contain string '$pattern'."
            msgOK="File contains string '$pattern'."/>
```

Validation chemistry

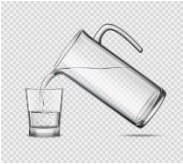
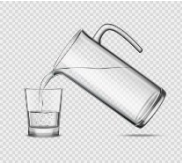
- Validation **report** = Σ Validation results
- Validation **result** = outcome of validating …

ONE resource against ONE constraint

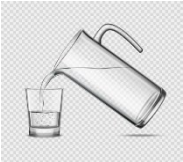
```
<gx:red filePath="C:/tt/greenfox/example-data/airports/airports.xml"
  msg="Airport elements should have EITHER @href OR content"
  constraintComp="ExprValueEmpty"
  constraintID="hrefOrChildren-empty"
  valueShapeID="hrefOrChildren"
  valueCount="2"
  exprLang="xpath"
  expr="//airport[@href and *]">
  <gx:valueNodePath>/airports/airport[1]</gx:valueNodePath>
  <gx:valueNodePath>/airports/airport[4]</gx:valueNodePath>
</gx:red>
```

Pouring waters of validity

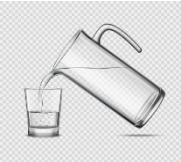
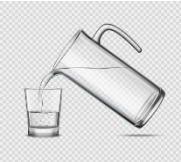
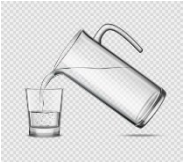

Validation of a file system tree against a greenfox schema: Given a file system tree and a greenfox schema, the validation results are the union of results of the validation of the file system tree against [redacted] in the greenfox schema.

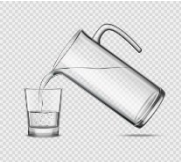



Validation of a file system tree against a shape: Given a file system tree and a shape in the greenfox schema, the validation results are the union of the results of the validation of [redacted] that are in the target of the shape.







Validation of a focus resource against a shape: Given a focus resource in the file system tree and a shape in the greenfox schema, the validation results are the union of the results of the validation of the focus resource against [redacted] declared by the shape.

Implementation

https://github.com/hrennau/greenfox

Apologies for the incomplete documentation – extension under construction.

Greenfox – a summary

- Goal: validation of a file system tree
- Based on XDM
- Powered by XPath and foxpath
- Inspired by SHACL
- Mediatypes hidden behind XDM node trees
- Resource boundaries hidden by foxpath navigation
- Producing structured information
- Extensible

Thank you.