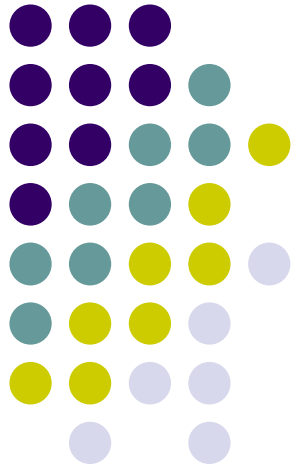
parsQube

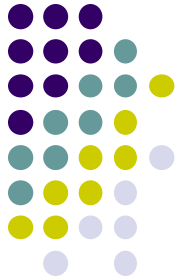
An introduction to Greenfox



A schema language describing file system contents - hands-on & brainfriendly

Hans-Jürgen Rennau, parsQube GmbH
Presented at Declarative Amsterdam 2020, October 8, 2020

File system tree validation



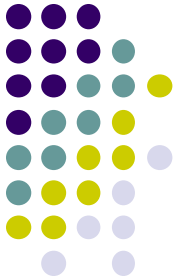
File system tree =
  a selected folder
    + all folders/files directly or indirectly contained

Validation =
  check conformance to a set of constraints („schema“)

Validation result =
  the outcome of one check:
    single resource checked against a *single constraint*

Validation report =
  collected validation results,
    mapped to something palatable

Why might you care?



- What we are used to:

  declarative validation of **single files** against schemas

  (XSD, RelaxNG, JSON Schema, CSV Schema, SHACL, …)

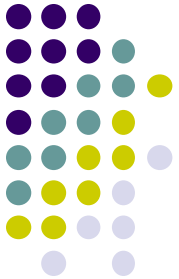
- Real interest: validity of systems, not individual files
- Single file: a tiny jigsaw piece in the picture of system validity
- File system trees are simply *larger parts of the picture*

  Examples:
  - A product to be shipped
  - A set of applications in use
  - Critical components of infrastructure
  - Data sources and assets
  - Complex test results

SAMPLE WORRIES

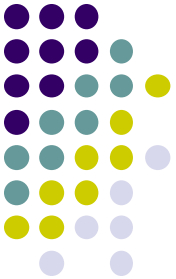
No file forgotten?
File versions correct?
Log files removed?
Documentation complete?
All translations included?
All links updated? Etc. etc.

Outline



- Getting started - hands-on impressions
- Big Picture - concepts
- Overview - available constraint types

Part 1: Getting started …



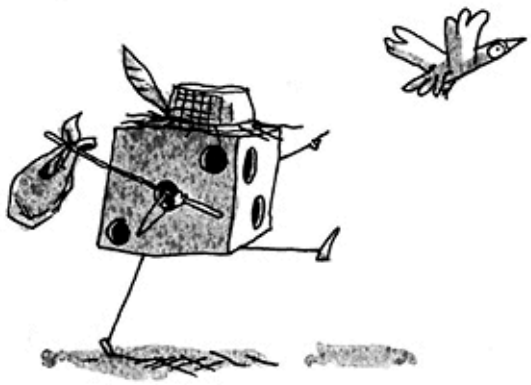
A guided tour:

„A TRIVIAL FILE SYSTEM TREE VALIDATED AGAINST A NON-TRIVIAL SCHEMA DEVELOPED IN SEVEN STEPS

Source of all airport data: **https://openflights.org/data.html**

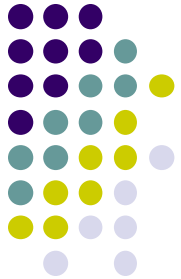
Jodle Jodle

(having made up his mind to participate
In the Greenfox tutorial at Decl. Amst. 2020)



(Jodle will join us, coming straight from the pencil of Cédric Philippe)
http://cedricphilippe.com/section_me.html

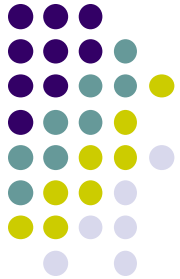
Part 2: Big picture



The key to understanding Greenfox is knowing

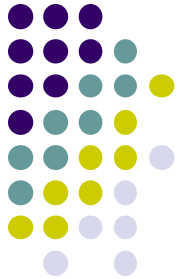
SEVEN CONCEPTS

SEVEN CONCEPTS



- Resources
- Constraints
- Shapes
- Target declarations
- Link definitions
- Results
- Reports

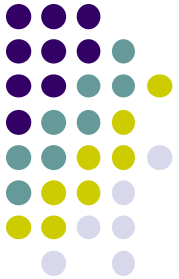
1. Resources



- Resources = files & folders

Now that was easy!

2. Constraints



- A constraint is a function:

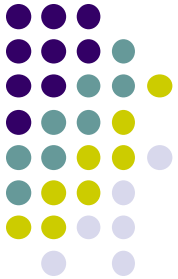
  Maps a resource to a validation result
  = (1) pass|failure (2) details

- Selection of the resource is *not* part of the constraint –
  that's the business of the containing shape

- Schema representation: XML element + attributes + children

```
<value exprXP="//airport/@id"
       minCount="1"    minCountMsg="Missing data: ID"
       distinct="true" distinctMsg="IDs not distinct"/>
```

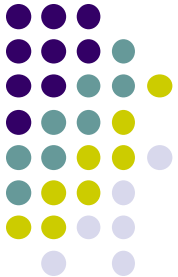
2. Constraints - *type + facet*



- Type, parameters, facet
  - The type is identified by the XML element name
  - The parameters are provided by attributes / child elements
  - The facet depends on a key parameter
- Example: Constraint #1
  - Type: Value
  - Parameters: exprXP, minCount, minCountMsg
  - Key parameter: minCount
  - Facet: ValueMinCount

```
<value exprXP="//airport/@id"
       minCount="1"    minCountMsg="Missing data: ID"
       distinct="true" distinctMsg="IDs not distinct"/>
```

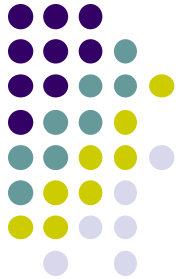
2. Constraints - *example*



- Example: Constraint #2
  - Type: Value
  - Parameters: exprXP, distinct, distinctMsg
  - Key parameter: distinct
  - Facet: ValueDistinct

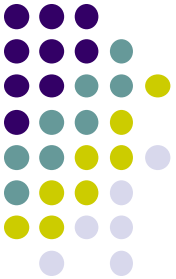
```
<value exprXP="//airport/@id"
       minCount="1"    minCountMsg="Missing data: ID"
       distinct="true" distinctMsg="IDs not distinct"/>
```

2. Constraints - *categorization*



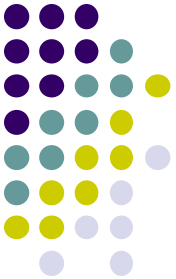
- Categorization:
  - Unary - targets **single resource**
    e.g. <value>, <valuePair>, <docTree>
  - Binary - targets a **pair of resources**
    e.g. <valueCompared>, <docSimilar>, <folderSimilar>
- Categorization:
  - Closed - **excludes impact** from other resources
    e.g. <value>, <valueCompared>
  - Open - **allows impact** from other resources
    e.g. <foxvalue>, <foxvaluePair>, <links>

2. Constraints - *constraint types*



Constraint type	Element	File (F) or Folder (D)	Unary/Binary (U|B) / Closed/Open (C|O)	Resource properties (P) or content (C)
FileDate	<fileDate>	F, D	U/C	P
FileName	<fileName>	F, D	U/C	P
FileSize	<fileSize>	F, D	U/C	P
FolderContent	<folderContent>	D	U/C	C
Mediatype	<mediatype>	F	U/C	C
DocTree	<docTree>	F	U/C	C
HyperdocTree	<hyperdocTree>	F, D	U/**O**	C
XsdValid	<xsdValid>	F	U/C	C
Value	<value>	F	U/C	C
ValuePair	<valuePair>	F	U/C	C
Foxvalue	<foxvalue>	F, D	U/**O**	C
FoxvaluePair	<foxvaluePair>	F, D	U/**O**	C
ValueCompared	<valueCompared>	F	**B**/C	C
FoxvalueCompared	<foxvalueCompared>	F, D	**B**/**O**	C
DocSimilar	<docSimilar>	F	**B**/C	C
FolderSimilar	<folderSimilar>	D	**B**/C	C
Link	<links>	F, D	U/**O**	*(depends)*
TargetSize	<targetSize>	F, D	U/**O**	*(depends)*
Conditional	<conditional>	F, D	*(depends)*	*(depends)*

3. Shapes

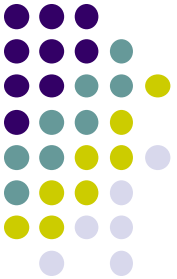


- Shape is two things:
  - Set of constraints
  - Target declaration
- Target declaration:

  „The constraints apply to these resources: (a selector)“
- Schema representation of a shape: <file>, <folder>
  - Element name: the kind of resources
  - Attributes: target declaration
  - Child elements: constraints

```
<file navigateFOX="*.FLAG">
    <targetSize minCount="0" minCountMsg="Missing FLAG file"/>
    <fileSize eq="10" eqMsg="FLAG file not empty"/>
</file>
```

4. Target declaration



- A target declaration is a function:

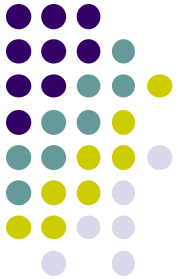
  Maps a resource to a set of resources

  * Input resource =
    a resource from the target of the parent shape
  * Output resources =
    contribution to the target of this shape

- Schema representation: attributes of <file> or <folder>

```
<folder navigateFOX="*\csv">...</folder>
<folder reflector1FOX="ancestor~::data" reflector2FOX="..\data-20201002">...</folder>
<file   navigateFOX="airports-*.json">...</file>
<file   hrefXP="//xs:import/@schemaLocation">...</file>
```

5. Link definition



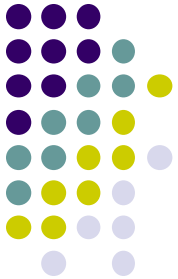
- A Link Definition is a function:

  Maps a resource to a set of resources

- Schema representation:
  - Either: <linkDef> element
  - Or: Attributes and child elements of a „link using element“

- Link using elements:
  - Shapes <file>, <folder>
  - Links constraint <links>
  - Binary constraints <valueCompared>, <docSimilar>, <folderSimilar> …
  - Hyperdoc constraint <hyperdocTree>

5. Link definition - *connectors*



Connector: foxpath

```
<linkDef name="myJSON"
         navigateFOX="fox-sibling($fileName, '\.xml', '.json')"/>
```

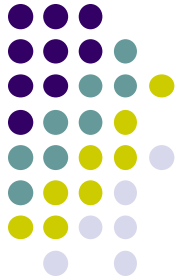
Connector: href-expression

```
<linkDef name="hrefAtts"
         hrefXP="//@href"/>
```

Connector: URI-expression

```
<linkDef name="href2JSON"
         uriXP="//href/replace(., 'json', 'xml')"/>
```

5. Link definition - *more connectors*



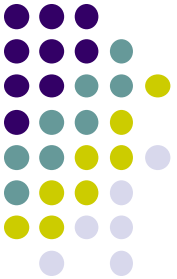
Connector: mirror

```
<linkDef name="mirror20201006"
         reflector1URI="${domain}"
         reflector2URI="${domain}\..\air.20201006"/>
```

Connector: URI-template

```
<linkDef name="jsonAirports"
         contextXP="/csv/record"
         uriTemplate="/air/countries/{country}/json/airport-{iata}.json">
    <templateVar name="country" valueXP="Country/lower-case(.)"/>
    <templateVar name="iata" valueXP="IATA/lower-case(.)"/>
</linkDef>
```

5. Link definition
- used by target declarations



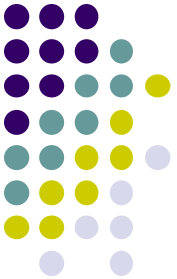
Link Definition **referenced** (@linkName)

```
<file linkName="myJSON">...</file>
```

Link Definition **local** (@navigateFOX, @hrefXP, @uriXP, ...)

```
<file navigateFOX="*.FLAG">...</file>

<file hrefXP="//href">...</file>

<file uriXP="//href/replace(., 'json', 'xml')"/>
```

5. Link definition
- used by binary constraints



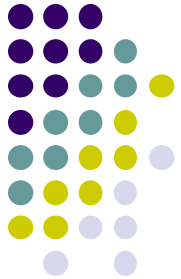
valueCompared

```
<valuesCompared linkName="myJSON" countTargetResources="1">
    <valueCompared expr1XP="/*/@country" minCount1="1"
                   expr2XP="//country"
                   cmp="eq"/>
</valuesCompared>
```

DocSimilar

```
<docSimilar linkName="schemaInitial">
    <ignoreValue kind="attribute" localName="timestamp"/>
</docSimilar>
```

5. Link definition - *constraints*



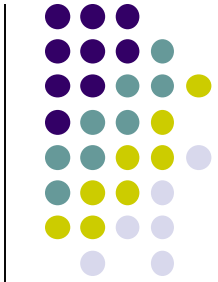
Links **resolvable** and yield **at least one** link target resource

```
<links linkName="hrefElems"
       resolvable="true"
       minCountTargetResources="1"/>
```

Exactly one link target resource

```
<links linkName="myXML"
       countTargetResources="1"/>
```

Intermezzo (interfoxo)



Note the the amazing fox – look at …

$greenfox/declarative-amsterdam-2020/the-amazing-fox/the-amazing-fox.txt
$greenfox/declarative-amsterdam-2020/tutorial-foxpath/tutorial-foxpath.txt

For example, node tree and file system navigation can be freely mixed, e.g.

```
fox "ancestor~::decl*//*.gfox.xml[\\*:docSimilar]"
```

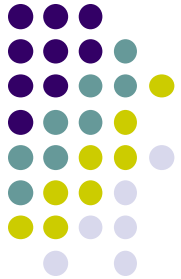
selects files based on their structured XML content. Or

```
fox "ancestor~::decl*//airport-*.json/jdoc(.)[\\iata eq 'WAT']"
```

selects files based on their structured JSON content.

NOTE: In Greenfox, the roles of / and \ are reversed. If you use **`fox`** with option **`–b`**, it behaves like Greenfox.

6. Results

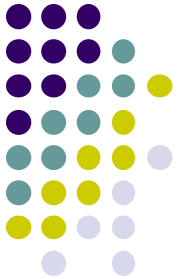


- Validation result =

Outcome of checking a single resource against a single constraint

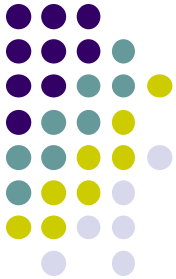
- XML element <red>, <green>
- Attributes and child elements …
  - Identify the resource
  - Identify the constraint type and facet
  - Constraint location in the schema
  - Constraint parameters
  - Observations

6. Result - *example*



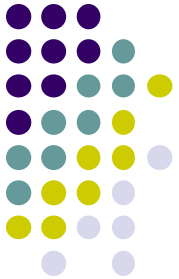
```
<gx:red file           = "/airports/index/airports-ireland.xml"
    constraintComp     = "ValueItemsDistinct"
    constraintPath     = "gx:values[1]/gx:value[1]/@distinct"
    resourceShapePath  = "/gx:greenfox[1]/gx:domain[1]/gx:folder[1]/gx:file[1]"
    distinct           = "true"
    valueCount         = "17"
    exprLang           = "xpath"
    expr               = "//airport/@id"
    quantifier         = "all">
    <gx:value nodePath="/airportsForCountry[1]/airport[4]/@id">600</gx:value>
    <gx:value nodePath="/airportsForCountry[1]/airport[5]/@id">600</gx:value>
</gx:red>
```

7. Report - *example*



```
G r e e n f o x   r e p o r t   s u m m a r y

greenfox: C:/tt/greenfox/declarative-amsterdam-2020/schema/air08.gfox.xml
domain:   C:/tt/greenfox/declarative-amsterdam-2020/data/air

#red:     1   (1 resources)
#green:   2518   (73 resources)

-----------------------------------------------
| Constraint Comp              | #red | #green |
|------------------------------|------|--------
| DocTreeClosed ..............  |    - |    128 |
| DocTreeMaxCount ............  |    - |   1152 |
| DocTreeMinCount ............  |    - |   1152 |
| FileSizeEq .................  |    - |      1 |
| FolderContentClosed ........  |    - |      1 |
| FolderContentMaxCount ......  |    - |      6 |
| FolderContentMd5 ...........  |    - |      1 |
| FolderContentMinCount ......  |    - |      8 |
| LinkResolvable .............  |    - |     13 |
| LinkTargetResourcesCount ...  |    - |     12 |
| LinkTargetResourcesMinCount   |    - |      7 |
| TargetCount ................  |    - |      5 |
| TargetMinCount .............  |    - |      2 |
| ValueCompairedValue1MinCount |    - |      3 |
| ValueComparedEq ............  |    - |      3 |
| ValueDatatype ..............  |    - |      3 |
| ValueEq ....................  |    - |      3 |
| ValueItemsDistinct .........  |    1 |      2 |
| ValueLt ....................  |    - |      3 |
| ValueMatches ...............  |    - |      3 |
| ValueMinCount ..............  |    - |      3 |
| ValuePairEq ................  |    - |      3 |
| ValuePairValue1MinCount ....  |    - |      3 |
| XsdValid ...................  |    - |      1 |
-----------------------------------------------

Red resources:
  F C:/tt/greenfox/declarative-amsterdam-2020/data/air/airports/index/airports-ireland.xml   (ValueItemsDistinct)
```

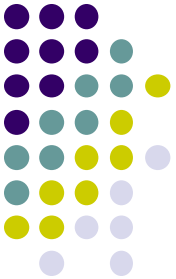
Part 3: Constraint types



Now you are ready to familiarize yourself with

CONSTRAINT TYPES

Constraint types

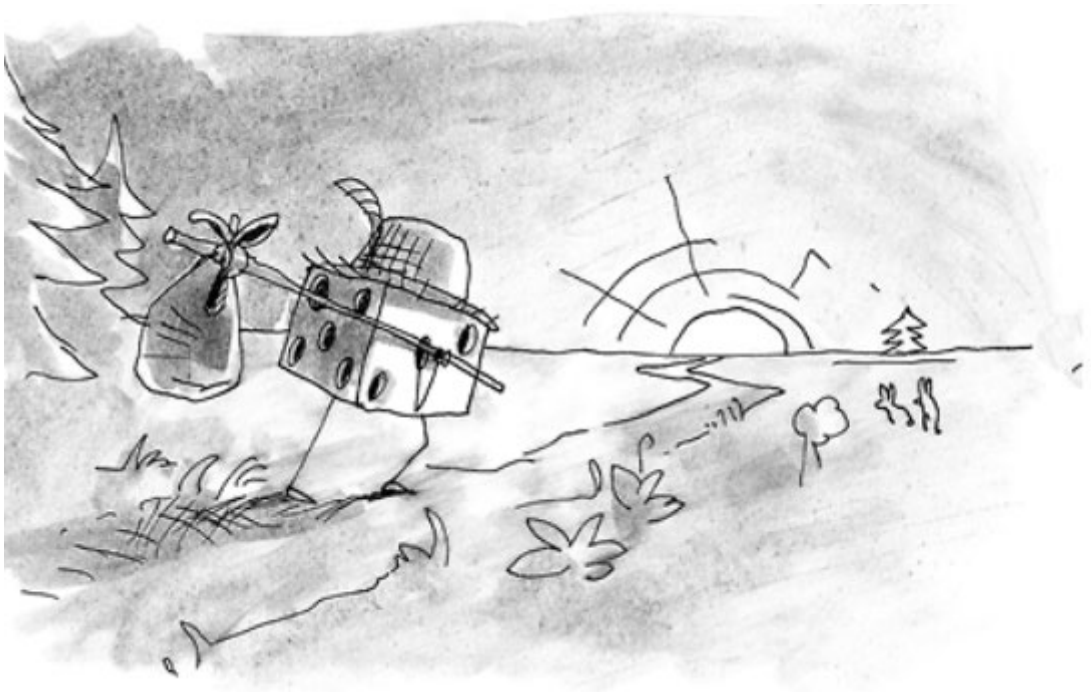


Constraint type	Element	File (F) or Folder (D)	Unary/Binary (U|B) / Closed/Open (C|O)	Resource properties (P) or content (C)
FileDate	<fileDate>	F, D	U/C	P
FileName	<fileName>	F, D	U/C	P
FileSize	<fileSize>	F, D	U/C	P
FolderContent	<folderContent>	D	U/C	C
Mediatype	<mediatype>	F	U/C	C
DocTree	<docTree>	F	U/C	C
HyperdocTree	<hyperdocTree>	F, D	U/**O**	C
XsdValid	<xsdValid>	F	U/C	C
Value	<value>	F	U/C	C
ValuePair	<valuePair>	F	U/C	C
Foxvalue	<foxvalue>	F, D	U/**O**	C
FoxvaluePair	<foxvaluePair>	F, D	U/**O**	C
ValueCompared	<valueCompared>	F	**B**/C	C
FoxvalueCompared	<foxvalueCompared>	F, D	**B**/**O**	C
DocSimilar	<docSimilar>	F	**B**/C	C
FolderSimilar	<folderSimilar>	D	**B**/C	C
Link	<links>	F, D	U/**O**	*(depends)*
TargetSize	<targetSize>	F, D	U/**O**	*(depends)*
Conditional	<conditional>	F, D	*(depends)*	*(depends)*

Important, but only mentioned

- `<focusNode>` - changing evaluation context
- Variable bindings in XPath and Foxpath (e.g. `$fileName`)
- Dealing with archives
- JSON, CSV, HTML, .txt (see also *demo-mediatype*)
- Schema context & schema parameters

Thank you, Jodle and all others, for your kind attention!



(Jodle returning to Prague, straight to the pencil of Cédric Philippe)
http://cedricphilippe.com/section_me.html