PROCESSING NON-XML SOURCES AS XML

XML Amsterdam

Alain Couthures - agenceXML

06/11/2015

TREES

XML, a tree with typed nodes

- Different node types:
  - DOCUMENT_NODE
  - ELEMENT_NODE
  - ATTRIBUTE_NODE
  - TEXT_NODE
  - COMMENT_NODE
  - …
- A restricted hierarchy

Trees everywhere

- XML, HTML
- JSON
- CSV
- XQuery
- C-family programs
- …

a grammar ==> a tree

A tree or a bunch of trees?

- Adding a document node for storing document-level information (filename, encoding, times, location, author,…)
- Accessible from any node using ownerDocument property

CSV as a tree

- Without header, an array of arrays
- With header, an array of maps
- Values not typed, usually

Model for Tabular Data and Metadata on the Web

W3C Working Draft 16 April 2015

http://www.w3.org/TR/2015/WD-tabular-data-model-20150416/

JSON as a tree

- A mix of arrays, maps and typed values
- Types for values limited to 'double', 'boolean' and 'string'
- Empty array, empty map and null
- Stand-alone atomic values
- JSON or Javascript?:
  regular expressions, dates, functions,...

JSON-LD: notations/semantics

- 3 notations for the same semantics
- Mix of data and metadata
- Need for the most convenient work structure
- Do not treat as JSON

XQuery as a tree

- XQueryX is an XML notation for XQuery including XPath
- Parsing XQuery into XQueryX then serializing back to XQuery:
  - a beautifier?
  - explicit notation: what about parenthesis? axes? …
- XSLTX as a full XML notation for XSLT?

C-family program as a tree

- Same problem as with XSLT:
  what is the best granularity?
- Comments and annotations are potentially interesting…
  (different namespaces analogy)
- Can be very useful to automatically modify program sources!

From a grammar to a tree

- ABNF grammar + indication for terms not rendered as element

  example:

  ```
  main = ^S ^E ^S
  E = float / ^E ^S (plus / minus) ^S float
  plus = ^"+"
  minus = ^"-"
  float = ^int ["." 3^int]
  int = 1*digit
  S = *^wsp
  ```

From a grammar to a tree

1 + 2.314 - 4567

becomes

```
<main>
   <float>1</float>
   <plus/>
   <float>2.314</float>
   <minus/>
   <float>4567</float>
</main>
```

IMPLEMENTING A MULTIPURPOSE DATA MODEL

JSON support in XPath 3.1

- map {} and array {} as constructors
- maps and arrays are functions to be called by just adding a parameter (key or position)
- ? operator as a shortcut

example: '$var?a?a1'
where $var contains {'a': {'a1': '1'}}

As XML, not into XML

- Extra node types for arrays and maps/entries
- Atomic typed value nodes, not just text nodes (null belongs to null-type)
- Absolutely no restriction on node types hierarchy
- Absolutely no restriction on node names

EXML Notation

- Serializer generates an XML-compatible string:
  - exml:document
  - exml:map / exml:entry, exml:array
  - exml:atom
  - exml:element, exml:attribute
  - …
- Parser generates initial tree structure back

Introducing "Fleur"

- Extended DOM Level 3 implementation
- Open source product in Javascript for browsers and Node.js

  https://github.com/AlainCouthures/Fleur

Fleur.Parser and Fleur.Serializer

- Fleur.DOMParser.prototype.parseFromString = function(s, mediatype, grammar)

- Fleur.Serializer.prototype.serializeToString = function(node, mediatype, indent)

Mediatypes already supported in Fleur

- "application/xml"
- "application/exml+xml"
- "text/csv"
- "application/json"
- "application/xquery" (partially)

Parser/Serializer options

- Parameters are serialized after the mediatype:
  text/csv;header=present
- Indentation as an option with different sub-options?

Adding new mediatype handlers

Fleur.Parser.Handlers ["..."] = function(node, s, config) { ... };

Fleur.Serializer.Handlers["..."] = function(node, indent, config) { ... };

where config is a Javascript object as parameter in mediatype

Extending XPath for non-XML

- New node tests:

  array(), map(), entry(), atom(type)

- Entry access with '?' prefix:

  map()/?a/map()/?a1

- Backquotes for non-XML names:

  `First Name`

EXAMPLES

JSON instance in XForms

```
<?xml-stylesheet type="text/xsl" href="../../xsltforms.xsl"?>
<html xmlns:xforms="http://www.w3.org/2002/xforms" xmlns:txs="http://www.agencexml.com/txs" xmlns="http://www.w3.org/1999/xhtml"
xmlns:xf="http://www.w3.org/2002/xforms">
        <head>
                <title>Process non-XML sources as XML</title>
                <xforms:model>
                        <xforms:instance mediatype="application/json">
                                [
                                        1,
                                        {
                                                Book: "On the Origin of Species",
                                        },
                                        {
                                                "First Name": "Charles"
                                        }
                                ]
                        </xforms:instance>
                </xforms:model>
        </head>
        <body>
                <table border="1">
                        <tr>
                                <td style="font-size: 24pt">Mediatype</td>
                                <td style="font-size: 18pt">application/json</td>
                        </tr>
                        <tr>
                                <td style="font-size: 24pt">Source</td>
                                <td>
                                        <pre>
                                                <xforms:output style="font-size: 14pt" value="serialize(/, 'application/json', 'yes')"/>
                                        </pre>
                                </td>
                        </tr>
                        <tr>
                                <td style="font-size: 24pt">eXML</td>
                                <td>
                                        <pre>
                                                <xforms:output style="font-size: 14pt" value="serialize(/, 'application/exml+xml', 'yes')"/>
                                        </pre>
                                </td>
                        </tr>
                        <tr>
```

JSON instance in XForms

Mediatype	application/json
Source	`[ 1, { Book: "On the Origin of Species" }, { "First Name": "Charles" } ]`

eXML	`<?xml version="1.0" encoding="UTF-8"?>`
	`<exml:document xmlns:exml="http://www.agencexml.com/exml">`
	`<exml:array>`
	`<exml:atom type="Q{http://www.w3.org/2001/XMLSchema}double">1</exml:atom>`
	`<exml:map>`
	`<exml:entry name="Book">`
	`<exml:atom type="Q{http://www.w3.org/2001/XMLSchema}string">On the Origin of Species</exml:atom>`
	`</exml:entry>`
	`</exml:map>`
	`<exml:map>`
	`<exml:entry name="First Name">`
	`<exml:atom type="Q{http://www.w3.org/2001/XMLSchema}string">Charles</exml:atom>`
	`</exml:entry>`
	`</exml:map>`
	`</exml:array>`
	`</exml:document>`

XPath	/array()/map()/?`First Name` = 'Charles'

CSV instance in XForms

Mediatype	text/csv;header=present;separator=%3B
Source	First Name;Birth Date Anthony;1996-05-31 Jeremy;1998-05-14 Aurélien;2002-06-08
eXML	<?xml version="1.0" encoding="UTF-8"?> <exml:document xmlns:exml="http://www.agencexml.com/exml"> <exml:array> <exml:element name="First Name"> <exml:atom type="Q{http://www.w3.org/2001/XMLSchema}string">Anthony</exml:atom> </exml:element> <exml:element name="Birth Date"> <exml:atom type="Q{http://www.w3.org/2001/XMLSchema}string">1996-05-31</exml:atom> </exml:element> </exml:array> <exml:array> <exml:element name="First Name"> <exml:atom type="Q{http://www.w3.org/2001/XMLSchema}string">Jeremy</exml:atom> </exml:element> <exml:element name="Birth Date"> <exml:atom type="Q{http://www.w3.org/2001/XMLSchema}string">1998-05-14</exml:atom> </exml:element> </exml:array> <exml:array> <exml:element name="First Name"> <exml:atom type="Q{http://www.w3.org/2001/XMLSchema}string">Aurélien</exml:atom> </exml:element> <exml:element name="Birth Date"> <exml:atom type="Q{http://www.w3.org/2001/XMLSchema}string">2002-06-08</exml:atom> </exml:element> </exml:array> </exml:document>
XPath	count(/array()) = 3

CSV instance with a key in XForms

Mediatype	text/csv;header=present;separator=%3B;key=0
Source	First Name;Birth Date Anthony;1996-05-31 Jeremy;1998-05-14 Aurélien;2002-06-08
eXML	<?xml version="1.0" encoding="UTF-8"?> <exml:document xmlns:exml="http://www.agencexml.com/exml"> <exml:element name="First Name"> <exml:map> <exml:entry name="Anthony"> <exml:element name="Birth Date"> <exml:atom type="Q{http://www.w3.org/2001/XMLSchema}string">1996-05-31</exml:atom> </exml:element> </exml:entry> <exml:entry name="Jeremy"> <exml:element name="Birth Date"> <exml:atom type="Q{http://www.w3.org/2001/XMLSchema}string">1998-05-14</exml:atom> </exml:element> </exml:entry> <exml:entry name="Aurélien"> <exml:element name="Birth Date"> <exml:atom type="Q{http://www.w3.org/2001/XMLSchema}string">2002-06-08</exml:atom> </exml:element> </exml:entry> </exml:map> </exml:element> </exml:document>
XPath	/`First Name`/map()/?*[name() = 'Jeremy']/`Birth Date` = '1998-05-14'

XQuery instance in XForms

Mediatype	application/xquery
Source	`/child::Config/child::Property[(attribute::name = "logfile")]`
XQueryX	<?xml version="1.0" encoding="UTF-8"?> <xqx:module xmlns:xqx="http://www.w3.org/2005/XQueryX"> <xqx:mainModule> <xqx:queryBody> <xqx:pathExpr> <xqx:rootExpr/> <xqx:stepExpr> <xqx:xpathAxis>child</xqx:xpathAxis> <xqx:nameTest>Config</xqx:nameTest> </xqx:stepExpr> <xqx:stepExpr> <xqx:xpathAxis>child</xqx:xpathAxis> <xqx:nameTest>Property</xqx:nameTest> <xqx:predicates> <xqx:equalOp> <xqx:firstOperand> <xqx:pathExpr> <xqx:stepExpr> <xqx:xpathAxis>attribute</xqx:xpathAxis> <xqx:nameTest>name</xqx:nameTest> </xqx:stepExpr> </xqx:pathExpr> </xqx:firstOperand> <xqx:secondOperand> <xqx:stringConstantExpr> <xqx:value>logfile</xqx:value> </xqx:stringConstantExpr> </xqx:secondOperand> </xqx:equalOp> </xqx:predicates> </xqx:stepExpr> </xqx:pathExpr> </xqx:queryBody> </xqx:mainModule> </xqx:module>
XPath	count(//xqx:equalOp[*/xqx:stringConstantExpr/xqx:value = 'logfile']) = 1

CONCLUSION

CONCLUSION

- A unique data model for them all
- Small extensions in XPath
- No cumbersome conversions
- Parsing/Serializing: always reversible
- All the power of the associated languages and tools