XML Overview, part 2

Norman Gray

Version 1.2, 2002/10/30

Contents

- Programming with XML
- Namespaces and architectures
- UR*
- RDF
- The future

Programming with XML

Contents

- Programming with XML
  - Parsers, languages and APIs
  - DOM
  - Programming with DOM
  - SAX *[. . .]*
- Namespaces and architectures
- UR*
- RDF
- The future

Parsers, languages and APIs

- There are numerous parsers, in Java, C, C++, Python, Perl, . . .
- See the Cover pages, `xml.coverpages.org`
- DOM and SAX are the main interfaces to XML parsers
- . . . but there are also other minimal ones
- XSLT and XSL-FO are languages to transform and format documents

DOM

- ‘Document Object Model’ allows you to wander round the tree
- All in memory (in principle)
- Allows arbitrarily complicated programmatic control over the DOM
- Doesn’t have to originate from an XML file! XML is not about angle-brackets!
- Java API: `org.w3c.dom.*`, supported in `javax.xml.*`
- Also `dom4j` from IBM, Xalan, `...`

Programming with DOM

```
import org.w3c.dom.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
public class SimpleDom {
    public static void main (String[] argv) throws Exception {
        Document doc =
            javax.xml.parsers.DocumentBuilderFactory.newInstance()
            .newDocumentBuilder().newDocument();
        Element el = doc.createElement("memo");
        doc.appendChild(el);
        Element kid = doc.createElement("from");
        kid.setAttribute("email", "norman");
        el.appendChild(kid);
        Transformer trans = TransformerFactory.newInstance().newTransformer();
        trans.transform(new DOMSource(doc),
                        new StreamResult(System.out));
    }
}
```

SAX

- Event model
- . . . so suitable for very large files
- Most suitable, *in general*, for formatting/searching
- . . . but not limited to that
- `www.saxproject.org`

Programming with SAX

```
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.helpers.XMLReaderFactory;

public class Poco extends DefaultHandler {
    public static void main (String[] args) throws Exception {
        XMLReader reader = XMLReaderFactory
          .createXMLReader("org.apache.xerces.parsers.SAXParser");
        Poco handler = new Poco();
        reader.setContentHandler(handler);
        reader.parse(args[0]);
    }
    public void startDocument() {
        System.out.print("Arf!");
    }
}
```

XSLT

- XSLT is the (main/standard) transformation language
- Powerful, and usable, though it looks a bit wierd to begin with
- XSL-FO (‘XSL Formatting Objects’) is a styling language; mostly for print
- CSS isn’t dead yet

Programming with XSLT, I

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:output method="html"/>

  <xsl:template match="/">
    <html>
      <head>
        <title>Memo from
          <xsl:apply-templates select="memo/from"/>
        </title>
      </head>
      <body>
        <xsl:apply-templates/>
      </body>
    </html>
  </xsl:template>
```

Programming with XSLT, II

```
  <xsl:template match="memo">
    <p><strong>From <xsl:apply-templates select="from"/></strong></p>
    <xsl:apply-templates select="p"/>
  </xsl:template>

  <xsl:template match="from">
    <xsl:value-of select="@email"/>
  </xsl:template>

  <xsl:template match="p">
    <p><xsl:apply-templates/></p>
  </xsl:template>
</xsl:stylesheet>
```

Programming with XSLT, III

Turns

```
<?xml version="1.0"?>
<memo>
  <from email="norman@astro.gla.ac.uk"/>
  <p>Hello, there</p>
  <p>How are you?</p>
</memo>
```

into

```
<html>
<head>
<title>Memo from
    norman@astro.gla.ac.uk</title>
</head>
<body>
<p>
<strong>From norman@astro.gla.ac.uk</strong>
</p>
<p>Hello, there</p>
<p>How are you?</p>
</body>
</html>
```

XPath

General syntax for specifying parts of a DOM tree. Not in XML syntax. Compact and powerful: regexp-ish.

`/memo`	top level element
`/memo/p`	all its (immediate) paragraph children
`p`	*all* the paragraphs
`/memo/p[2]`	the third paragraph child (count from zero)
`from/@email`	`from` elements' `email` attributes

Infoset/DOM: abstract data structures

- The *Infoset* is a definition of the collection of information items that are available as the result of an XML parse
- For example, it says that information about the order of elements is available, but the order of attributes isn't
- Really just a vocabulary for standards writers, however, and isn't directly/explicitly manipulated in any API.
- The DOM isn't just a way of getting at an XML parse: you can view it as a perfectly general API for getting at structured information
- Standards like XPath, XQuery, XSLT operate on the DOM/Infoset
- An XML file is just *one* way of stocking a DOM

Namespaces and architectures

Contents

- Programming with XML
- Namespaces and architectures
  - Namespaces
  - . . . as architectures
  - Interpellating* documents
- UR*
- RDF
- The future

Namespaces

- Namespaces allow you to link element types to ‘owners’, and hence to syntax and semantics
- A bit like DTDs, but without syntax checking
- Named by URIs, which are opaque – not dereferencable
- Default namespace; but watch attributes

```
<myformat>
  <mytitle>Pretty picture</mytitle>
  <x:hdx xmlns:x='http://www.starlink.ac.uk/HDX'>
    <x:ndx x:uri="http://example.edu/myfile.fits" title="awww"/>
  </x:hdx>
  <html xmlns="http://www.w3.org/TR/REC-xhtml">
    <head><title>HTML title</title></head>
    <body><p><a href="http://www.nowhere.com"></p></body>
  </html>
</myformat>
```

. . . as architectures

- Like (old) architectures: ideal for ‘loose collaboration’ between originators of DTDs
- Extract specialised ‘view’ of an XML document
- ```
  <myformat
    xmlns:n=’http://www.starlink.ac.uk/HDX’>
    <link n:name=’data’>
      http://example.edu/myfile.fits</link>
  </myformat>
  ```
- AF-NG
  ```

Interpellating* documents

- Can automatically extract ‘our’ syntax from the mess of stuff we don’t know or care about
- Namespace stuff can be hidden in DTD, to some extent
- Simple but *generic* transformation
- Lowers barriers to using our software

```
<myformat
  xmlns:n='http://www.starlink.ac.uk/HDX'>
  <link n:name='data'>
    http://example.edu/myfile.fits</link>
</myformat>
```

to

```
  <data uri="http://example.edu/myfile.fits"/>
```

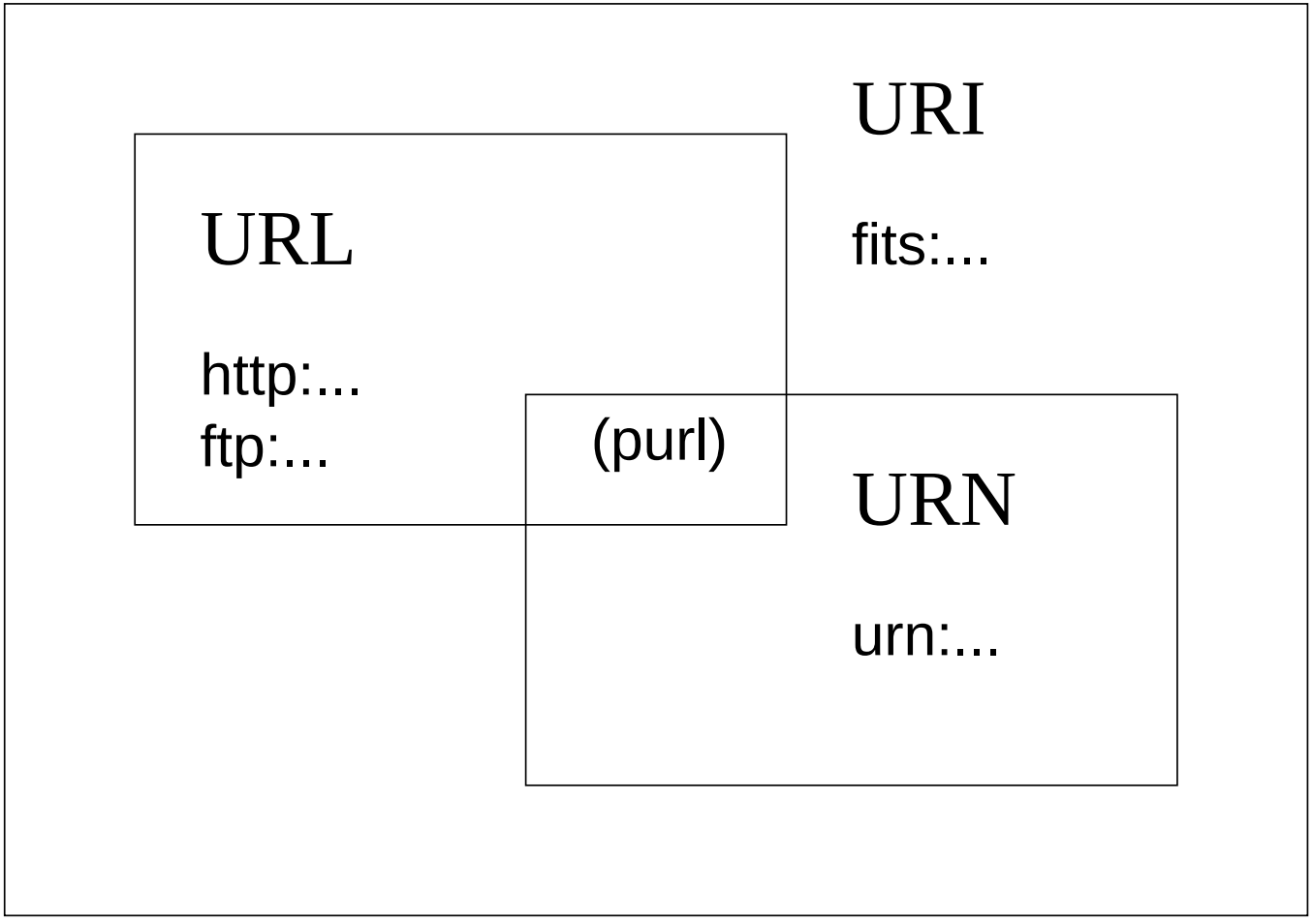
* look it up

UR*

Contents

- Programming with XML
- Namespaces and architectures
- UR*
  - URIs, URNs and URLs
  - URI vs. URL vs. URN
- RDF
- The future

URIs, URNs and URLs



URI vs. URL vs. URN

- URIs are general *names* for resources (RFC 2396)
- URLs are URIs with *location* info
- URNs are URIs with *“an institutional commitment to persistence”*
- . . . either RFC 2141 (`urn:...`), or PURLs (`http://purl.oclc.org/OCLC/PURL/INET96`)
- So (URI or URN) to URL needs *resolver* service
- URIs are either ‘hierarchical’ or ‘non-hierarchical’; former are slightly manipulable (‘up’, ‘relative’, etc, requiring resolution), latter are opaque
- Needn’t correspond to a single file
- Fragments resolved at client

RDF

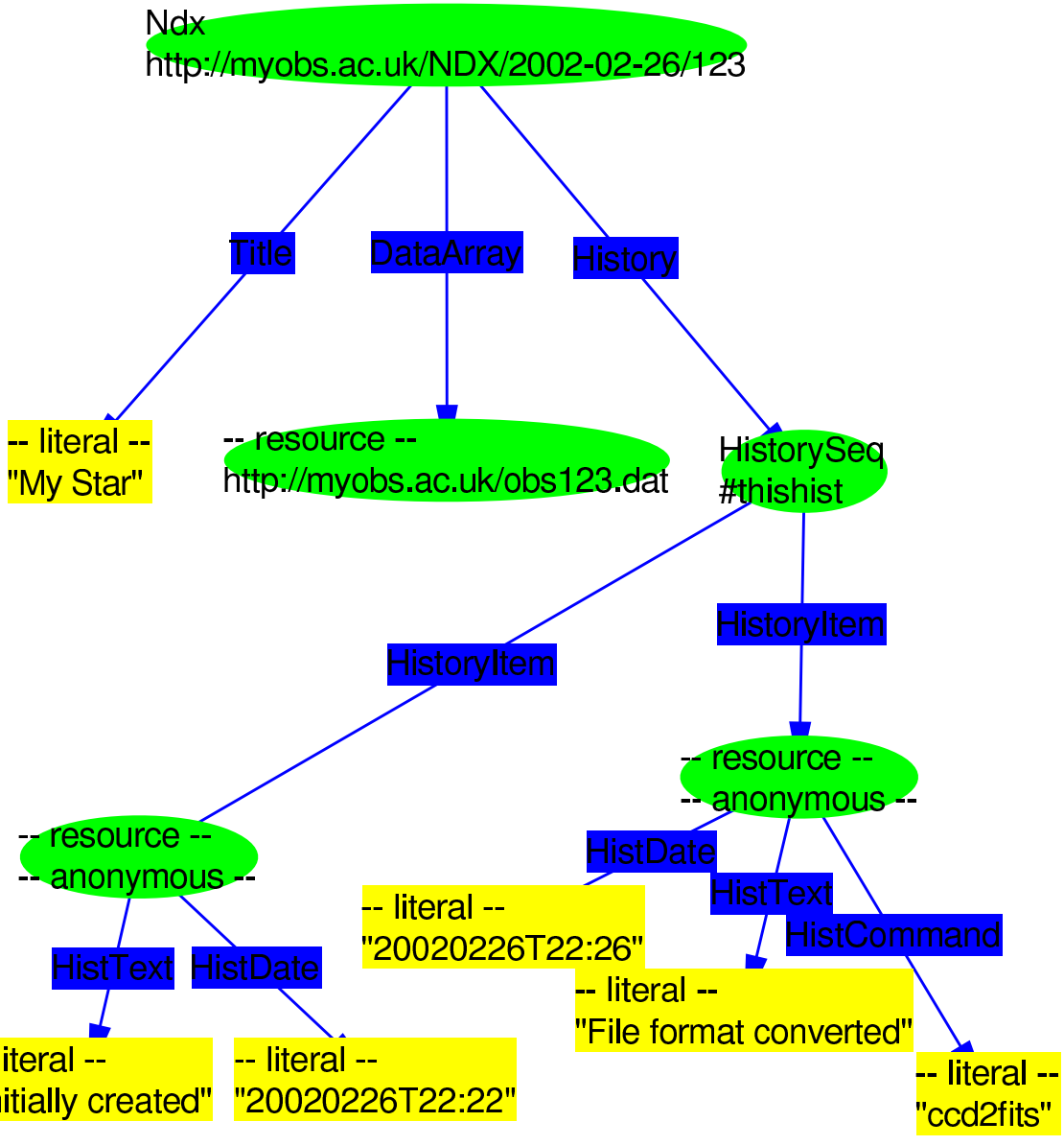
Contents

- Programming with XML
- Namespaces and architectures
- UR*
- RDF
  - RDF for metadata
  - RDF illustrated
  - RDF data model
- The future

RDF for metadata

- RDF is a *framework* for metadata
- The Right Thing, even though there are few tools as yet
- An *RDF Schema* is a vocabulary, or *ontology*
- XML notation, ‘Notation 3’, others. . .
- Hasn’t yet found much traction, though the user story is very attractive
- The Semantic Web will take over the world (maybe)
- Should be easy to add *post hoc*

RDF illustrated



RDF data model

- RDF data model: ‘resources have properties which are resources’
- More primitive than XML (a directed graph rather than a tree)
- . . . is distinct from, and independent of, RDF-for-metadata: more generic idea, clarifying

The future

Many more questions than answers

- XML 1.1 has only minor changes – the fight about XML 2.0 hasn't even started yet
- Will XML get bigger or smaller?
- Will XML Schemas take over the world?
- DOM is a bit clunky: will it survive?
- Will architectures take off?

Perhaps TAG holds the answers; as long as we acquire XML-Fu we should remain in sympathy with changes