XML Overview, part 1

Norman Gray

Revision 1.4, 2002/10/30

Contents

- The who, what and why
- XML Syntax
- Programming with XML
- Other topics
- The future

`http://www.astro.gla.ac.uk/users/norman/docs/`

The who, what and why

Contents

What is XML?

- XML is ‘eXtensible Markup Language’
- XML is SGML`--`: SGML is ‘Standard Generalised Markup Language’, very robust, very large-scale
- `http://www.w3.org/TR/1998/REC-xml-19980210`, and `http://xml.coverpages.org`
- Standardized markup, intended to be easy to parse, and easy to navigate around
- Strongly hierarchical, but only sort-of object-orientated
- Supports two paradigms: XML as documents, and XML as database
- With the syntactic foundations sorted out, it’s easy (-ish) to add further standards which add semantics
- But XML is now turning into ++(`--`SGML)

(but what about HTML?)

- HTML *is* an SGML application
- . . . in the sense that HTML 2–4 were defined as SGML DTDs
- . . . and even though most browsers let you break most of the rules
- Set of elements (‘p’, ‘table’, ‘h1’, . . .) is useful but fixed
- ‘Extensible’ means XML allows you to define your own vocabulary of elements – defining a new syntax
- Semantics – the meaning – is separate, and that’s what applications add, using DOM, XSLT, or whatever

Why is XML?

- . . . because SGML is too hard, or too big, or too eighties
- . . . because writing robust parsers is boring
- . . . because validation makes life easier for processors (and their authors)
- . . . because a strongly hierarchical way of representing information is generally natural and useful, and particularly useful to us, used to using NDF, HDS, FITS

Who is XML?

- W3C, `www.w3.org`: the World Wide Web Consortium, which issues Drafts and Recommendations
- W3C is pay-to-play, and most of the big corporations are playing (not too many fouls); but so are other organisations, including RAL
- Plus RFCs for things like HTTP, URIs
- Plus community standards, like SAX
- `xml-dev`, XMLDeviant

XML Syntax

Contents

Tags and elements

```
<memo>
   <from email="norman@astro.gla.ac.uk"/>
   <p>Hello, there</p>
</memo>
```

- Tags versus elements, and empty elements
- Attributes versus element content
- Comments: `<!-- stuff without double-hyphens-->`
- Escaping: `&amp;`, `&lt;`, `&gt;`, or the blunt instrument of `<!CDATA[anything]]>`
- All Unicode, including element names
- Whitespace rules are complicated but unsurprising; if you care, read the XML 1.0 rec.

Well-formed XML

- All elements closed
- No overlapping elements: <b><i>forbidden</b></i>
- Attribute names are unique within a tag, and their values have quotes: <el att="value">
- Only one top-level element
- Addresses much of the problem, and DTDs solve much of the rest

DTD syntax

```
<!ELEMENT memo (from?, p+)>
<!ELEMENT from EMPTY>
<!ELEMENT p (#PCDATA)>
<!ATTLIST from
    email CDATA #IMPLIED>
```

- . . . doesn’t look too pretty, but it does the job
- Still heavily used
- Will probably last a long time

XML Schema syntax

- The current W3C-blessed syntax
- Written in XML instance syntax; rather verbose
- Has a more elaborate set of types, and can specify more elaborate constraints than DTD syntax is capable of
- . . . but not everything
- Popular with database folk
- Less ubiquitous application support, but politically important that it succeeds

Relax NG

- Community standard (from James Clark)
- Non-XML syntax, but readable
- Extensible
- Might well take off

Programming with XML

Contents

- The who, what and why
- XML Syntax
- Programming with XML
  - Parsers, languages and APIs
  - DOM
  - Programming with DOM
  - SAX *[. . .]*
- Other topics
- The future

Parsers, languages and APIs

- There are numerous parsers, in Java, C, C++, Python, Perl, . . .
- Numerous editors
- 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>
```

Other topics

Contents

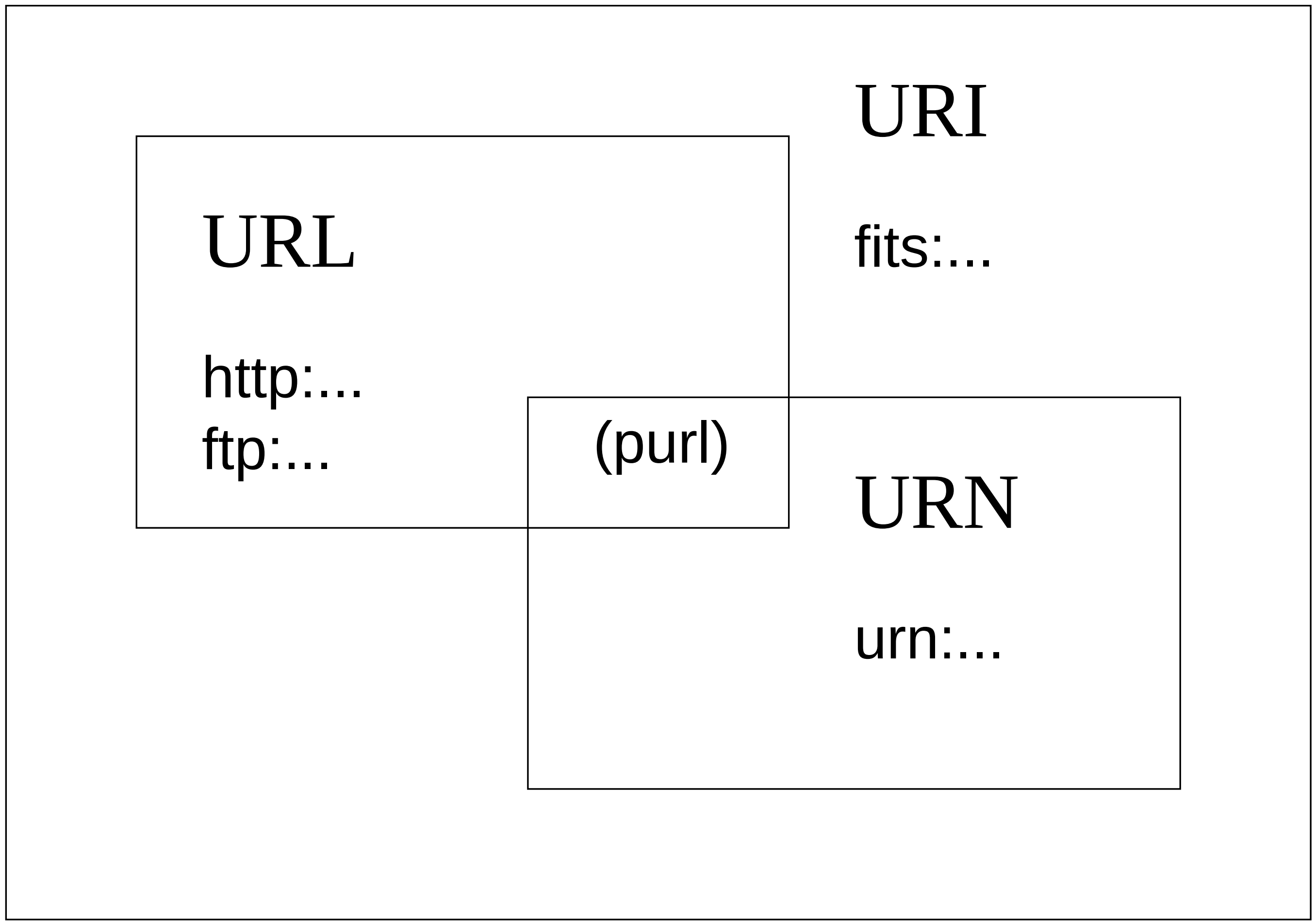
- The who, what and why
- XML Syntax
- Programming with XML
- Other topics
  - Namespaces
  - URIs, URNs and URLs
  - URI vs. URL vs. URN
- The future

Namespaces

- A way of keeping vocabularies apart from each other

-
  ```
  <html><title>Example</title>
    <p>Here is a FITS file</p>
    <hdx:hdx xmlns:hdx="http://www.starlink.ac.uk/HDX">
      <hdx:ndx hdx:uri="file:myfile.fits" title="My title"/>
    </hdx:hdx>
    <p>Wasn’t that exciting</p>
  </html>
  ```
- It’s basically that simple, but there are gotchas to do with default namespaces

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”*

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 Schemas take over the world?
- DOM is a bit clunky: will it survive?

But there are emerging principles which should keep every-one in step.