XML and perl

Did you every fly a kite in bed?
Did you ever walk with ten cats on your head?
...
If you never did, you should,
these things are fun, and fun is good.
-- *Dr. Seuss*

Some XML tricks with Perl

XML will enable universal data interoperability by providing a vender independant, OSI level 7 presentation layer, semantic capable, interoperable data format.

Unfortunately, this works best at the "slideware" level.

Beyond the hype, XML does provide many really useful things - but the devil is in the details, or specifically the support tools. Perl has come a long way in the last year to providing some really useful XML modules.

We will briefly discuss XML, and maybe xml schemas XPATH, DTDs or XSLT. Then we'll see a few XML modules in action.

What is XML?

- Andy's Answer: *A syntax to serialize data structures*
- XML looks like HTML
  - Conceptually very different
- Fairly simple spec
  - “Tons” of associated specs
- Standards mostly controlled by fairly sane people
- Broad support from open source to MS

Simple XML example

```
<simple_xml_example    easyness="high">
   <e1>
      text
      <content att1="1" att2="2"></content>
      more text
      <content    att2="2" att1="1"  />
   </e1>
   <e2/>
</simple_xml_example>
```

More compilated example

- File format for StarOffice 6.0 (ie openoffice.org) is XML based
- Text sections represented in XML + Images are zipped.
- Used the following document

-
  ```
  <H1>
        Just another OpenOffice.org hacker
  </H1>
  ```

More complicated example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE office:document-content
    PUBLIC "-//OpenOffice.org//DTD OfficeDocument 1.0//EN"
    "office.dtd">
<office:document-content
    xmlns:office="http://openoffice.org/2000/office"
    xmlns:text="http://openoffice.org/2000/text"
    xmlns:fo="http://www.w3.org/1999/XSL/Format"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:math="http://www.w3.org/1998/Math/MathML"
    office:class="text"
    office:version="1.0">
    <office:script/>
```

Some XML continued...

```
  <style:font-decl
          style:name="Albany"
          fo:font-family="Albany"
          style:font-family-generic="swiss"
          style:font-pitch="variable"/>
  </office:font-decls>
  <office:automatic-styles/>
  <office:body>
      <text:sequence-decls>
          <text:sequence-decl
              text:display-outline-level="0"
              text:name="Text"/>
      </text:sequence-decls>
      <text:h
          text:style-name="Heading 1"
          text:level="1">Just another OpenOffice.org hacker
      </text:h>
  </office:body>
</office:document-content>
```

Important XML Concepts:
XML is just a data structure

- XML is only syntax not *semantics*
- XML subspecs such as XHTML, SOAP, staroffice file format, define semantics

Important XML Concepts: *Validity versus Well-formed*

- An XML file which meets the spec is *well-formed*
- An XML file which is in accordance with its DTD or schema is *valid*
- XML does not require DTDs (unlike SGML)

Important XML Concepts:
XML info-set

- Info-set equivalent documents
- <a b=”c” d=”e”>f<g/>h&lt;</a>

-
  ```
  <a    d   =   “e”
        b = “c”     >f<g
                        ></g
                           >h&08;</a   >
  ```

Important XML Concepts: *XML info-set*

- Attribute order
- Spaces in tags
- CDATA sections
- Entities, quoted elements
- Name spaces

Good things about XML

- Text Format
  - Human viewable, editable, and *debuggable*
- Separates *Data-processing* from *Data-validation*
- Unicode throughout
  - Uses UTF-8 (not broken Windows and Java style BMP) encoding
- Intrinsically extensible

Bad Things About XML: *Confusing Tools*

- Standard API's are *painful*

```
my $nodes = $doc->getElementsByTagName ("CODEBASE");
my $n = $nodes->getLength;

for (my $i = 0; $i < $n; $i++)
{
    my $node = $nodes->item ($i);
    my $href = $node->getAttributeNode ("HREF");
    print $href->getValue . "\n";
}
```

Bad Things About XML: *Broken Tools*

- MSXML didn't work at all until version 3. It still doen't handle external entities properly
- No Debuggers for XML parsers. Now activestate has a visual XSLT
- No good tools for writing DTDs and schemas. So most people don't bother.

Bad Things About XML: *Bloated Formats*

$$F(y) = \frac{1}{2\pi}\int_{-\infty}^{\sqrt{y}} \left(\sum_{k=1}^{n} \sin^2 x_k(t) \right) f(t)\,dt$$

- Example: *MathML*

```
<?xml version="1.0" encoding="iso-8859-1"?>
<math xmlns="http://www.w3.org/1998/Math/MathML">
<mi>F</mi>
<mrow>
<mo>(</mo> <mi>y</mi> <mo>)</mo>
</mrow>
<mo>=</mo>
<mfrac>
<mn>1</mn> <mrow> <mn>2</mn> <mi>&pi;</mi> </mrow>
</mfrac>
<msubsup>
<mo>&Integral;</mo>
<mrow> <mo>-</mo> <mo>&infin;</mo> </mrow>
<msqrt> <mi>y</mi> </msqrt>
</msubsup>
```

Bad Things About XML: *Bloated Formats*

- Example: *SOAP*

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
<ns1:execute xmlns:ns1="urn:facerec-service"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
<command-list xmlns:ns2="http://schemas.xmlsoap.org/soap/encoding/
xsi:type="ns2:Array"
xmlns:ns3 = "urn:facerec-types"
ns2:arrayType="ns3:Command[10]">
<item xsi:type="ns3:UploadImageCommand">

...
```

Bad Things About XML: *Attribute / Entity confusion*

- When inventing an XML data format: what goes in entities and what goes in attributes

-
  ```
  <comparison>
      <source>A</source>
      <target>B</target>
      <score>10</score>
  </comparison>
  ```
- *or*

  ```
  <comparison source=”A” target=”B”
                          score=”10”/>
  ```

XML Advice

- *Don't hand parse XML*

- Be careful hand creating XML
  - need to “quote”, < , > , &

- Don't use XML::DOM
  Don't use XML::Parser

Modules: XML::Simple

- Use XML::Simple
  $xml_struct = XMLin(xml file or string)
    gives a hash of arrays data structure

  $xml_string = XMLout ($xml_structure)
- Good for small XML documents.
- Lots of options to control how structure is created. *I don't like the default options*

Modules: XML::Twig

- Build parse trees of parts of documents

```
use XML::Twig;

my $t= XML::Twig->new();
$t->parse(
'<d><tit>title</tit><para>para1</para><para>p2</para></d>');
my $root= $t->root;
$root->set_gi( 'html');                # change doc to html
$title= $root->first_child( 'tit');    # get the title
$title->set_gi( 'h1');                 # turn it into h1
my @para= $root->children( 'para');    # get the para children
foreach my $para (@para)
  { $para->set_gi( 'p'); }             # turn them into p
$t->print;                             # output the document
```

Modules: XML::XPath

- Xpath is like a *regexp* language for XML

```
use XML::XPath;
use XML::XPath::XMLParser;

my $xp = XML::XPath->new(filename => 'test.xhtml');

my $nodeset = $xp->find('/html/body/p');
          # find all paragraphs

foreach my $node ($nodeset->get_nodelist) {
    print "FOUND\n\n",
        XML::XPath::XMLParser::as_string($node),
        "\n\n";
}
```

Modules: XML::Xerces

- If you need a full power XML parser that does W3C schema, DTDs and all the rest
- Unfortunately has the old painful DOM API

Future of XML and Perl

- XML will not be well used until there is good *in-language* support (à la Perl regexps)
- It must be easier to use XML than to invent an *ini style file format
- Require “under the covers” caching
- Perl XML support is too scattered -we need less and better modules.
- The *Perl* way is/should be to make it easy to right thing, the *Java* way is to forbid the wrong thing. We must *perlize* XML.