ODF Programmability – What we need & What we have

Robert Weir
Software Architect
IBM Software Group
robert_weir@us.ibm.com
http://www.robweir.com/blog

© 2007 IBM Corporation

What we had before

- DOC/XLS/PPT
- Proprietary binary formats
- Documentation incomplete and outdated
- An opaque format distorts application development
  - Leads to “inside out” style
  - Put code in the document
  - Tied to Windows
  - Tied to client
  - Hard to manage
  - Vector for viruses
  - Code doesn't live where the documents live

The potential

- ODF – a platform and application neutral office file format
- Document data is no longer trapped in proprietary black box binaries
- Transparent format fosters external manipulation of documents
- This can lead to a “golden age” of document processing, both client and server side, with much innovation

- “We have it in our power to create the world over again” -- Thomas Paine

More than just editors

(20 Prototypical Scenarios)

1. Interactive creation in an a heavy-weight client application
2. Interactive creation in a light-weight web-based application
3. Collaborative (multi-author) editing
4. Automatic creation in response to a database query (report generation)
5. Indexing/scanning of document for search

20 Prototypical App Dev Scenarios

6. Scanning by anti-virus
7. Other types of scanning, perhaps for regulatory compliance, legal or forensic purposes
8. Validation of document, to specifications, house style guidelines, accessibility best practices, etc.
9. Read-only display of document on machine without the full editor (viewer)
10. Conversion of document from one editable format to another

20 Prototypical App Dev Scenarios

11. Conversion of document into a presentation format, such as PDF, PS, print or fax
12. Rendering of document via other modes such as sound or video (DAISY Talking Book)
13. Reduction/simplification of document to render on a sub-desktop device such as cell phone or PDA.
14. Import of data from an office document into a non-office application, i.e., import of spreadsheet data into statistical analysis software.
15. Export of data from a non-office application into an office format, such as an export of a spreadsheet from a personal finance application.

20 Prototypical App Dev Scenarios

16. Application which takes an existing document and outputs a modified version of that presentation, e.g., fills out a template, translates the language, etc.
17. Software which adds or verifies digital signatures on a document in order to control access (DRM)
18. Software which uses documents in part of a workflow, but treats the document as a black box, or perhaps is aware of only basic metadata.
19. Software which treats documents as part of a workflow, but is able to introspect the document and make decisions based on the content.
20. Software which packs/unpacks a document into relational database form.

The Problem

- 700+ page ODF Specification
- No objections to it as a specification – it is what it needs to be
- Written from the perspective of word processor implementors
- Too much to ask the average app developer to master

Analogy with XML -- Who actually reads this stuff?

3.3.3 Attribute-Value Normalization

Before the value of an attribute is passed to the application or checked for validity, the XML processor MUST normalize the attribute value by applying the al
other method such that the value passed to the application is the same as that produced by the algorithm.

1. All line breaks MUST have been normalized on input to #xA as described in **2.11 End-of-Line Handling**, so the rest of this algorithm operates on text
2. Begin with a normalized value consisting of the empty string.
3. For each character, entity reference, or character reference in the unnormalized attribute value, beginning with the first and continuing to the last, do th
   - For a character reference, append the referenced character to the normalized value.
   - For an entity reference, recursively apply step 3 of this algorithm to the replacement text of the entity.
   - For a white space character (#x20, #xD, #xA, #x9), append a space character (#x20) to the normalized value.
   - For another character, append the character to the normalized value.

If the attribute type is not CDATA, then the XML processor MUST further process the normalized attribute value by discarding any leading and trailing space
replacing sequences of space (#x20) characters by a single space (#x20) character.

Note that if the unnormalized attribute value contains a character reference to a white space character other than space (#x20), the normalized value contai
(#xD, #xA or #x9). This contrasts with the case where the unnormalized value contains a white space character (not a reference), which is replaced with a s
normalized value and also contrasts with the case where the unnormalized value contains an entity reference whose replacement text contains a white spac
processed, the white space character is replaced with a space character (#x20) in the normalized value.

All attributes for which no declaration has been read SHOULD be treated by a non-validating processor as if declared **CDATA**.

What is really used is SAX

endElement

```
public void endElement(java.lang.String uri,
                       java.lang.String localName,
                       java.lang.String qName)
                throws SAXException
```

Receive notification of the end of an element.

The SAX parser will invoke this method at the end of every element in the XML document; there will be a corresponding `startElement` e

For information on the names, see startElement.

Parameters:

`uri` - the Namespace URI, or the empty string if the element has no Namespace URI or if Namespace processing is not being perfo
`localName` - the local name (without prefix), or the empty string if Namespace processing is not being performed
`qName` - the qualified XML name (with prefix), or the empty string if qualified names are not available

Throws:

`SAXException` - any SAX exception, possibly wrapping another exception

And DOM

```
public Node getParentNode();

public NodeList getChildNodes();

public Node getFirstChild();

public Node getLastChild();

public Node getPreviousSibling();

public Node getNextSibling();

public NamedNodeMap getAttributes();

public Document getOwnerDocument();

public Node insertBefore(Node newChild,
                         Node refChild)
                         throws DOMException;

public Node replaceChild(Node newChild,
                         Node oldChild)
                         throws DOMException;
```

The Challenge

We need an ODF API that exposes a higher level abstraction of ODF to application developers, so they can quickly become productive with ODF processing without having to master a 700 page specification

“Create a loan amortization spreadsheet in 30 lines of code”

Odfpy

ODFPY - Python API and tools

Odfpy aims to be a complete API for OpenDocument in Python. Unlike other more convenient APIs, this one is essentially an abstraction layer just above the XML format. The main focus has been to prevent the programmer from creating invalid documents. It has checks that raise an exception if the programmer adds an invalid element, adds an attribute unknown to the grammar, forgets to add a required attribute or adds text to an element that doesn't allow it.

These checks and the API itself were generated from the RelaxNG schema, and then hand-edited. Therefore the API is complete and can handle all ODF constructions, but could be improved in its understanding of data types.

Take a look at the **api-for-odfpy.odt** manual. Most of the manual is also generated from the RelaxNG schema.

In addition to the API, there are also a few scripts:

mailodf - Email ODF file as HTML archive
odf2mht - Convert ODF to HTML archive
odf2war - Convert ODF to KDE web archive
odflint - Check ODF file for problems
odfmeta - List or change the metadata of an ODF file
odfoutline - Show outline of OpenDocument
odfuserfield - List or change the user-field declarations in an ODF file
xml2odf - Create OD? package from OpenDocument in XML form

Download **Odfpy-0.5.tgz**.

Low Level, close to the XML
Maps validity errors into runtime exceptions.

```
from odf.opendocument import OpenDocumentText
from odf.style import Style, TextProperties
from odf.text import H, P, Span

doc = OpenDocumentText()
# Styles
s = textdoc.styles
h1style = Style(name="Heading 1", family="paragraph")
h1style.addElement(TextProperties(attributes={'fontsize':"24pt",'fontweight':"bo
ld" }))
s.addElement(h1style)
# An automatic style
boldstyle = Style(name="Bold", family="text")
boldprop = TextProperties(fontweight="bold")
boldstyle.addElement(boldprop)
textdoc.automaticstyles.addElement(boldstyle)
# Text
h=H(outlinelevel=1, stylename=h1style, text="My first text")
textdoc.text.addElement(h)
p = P(text="Hello world. ")
boldpart = Span(stylename="Bold",text="This part is bold. ")
p.addElement(boldpart)
p.addText("This is after bold.")
textdoc.text.addElement(p)
textdoc.save("myfirstdocument.odt")
```

odf4j

```
import org.openoffice.odf.spreadsheet.SpreadsheetDocument;
import javax.swing.table.TableModel;
import javax.swing.table.DefaultTableModel;
import java.sql.ResultSet;
import java.sql.Statement;
import java.io.OutputStream;
//...

public class MyClass {

public void createDocument(OutputStream out, String uri) {
SpreadsheetDocument doc=new SpreadsheetDocument();

// do database query
// ...

// create table model
TableModel model=new DefaultTableModel();

// fill model with result of database query
//...

// create Spreadsheet in the SpreadsheetDocument based on the data in the model
doc.addSpreadsheet("data",model);

// the SpreadsheetDocument constructor created a document with one empty Spreadsheet
// named "Sheet1" this Spreadsheet is not needed anymore, so we delete it
doc.deleteSpreadsheet("Sheet1");

// store the document
doc.save(out,uri);
}
// ...
}
```

Part of OpenOffice.org Toolkit project.
Still early.

AODL

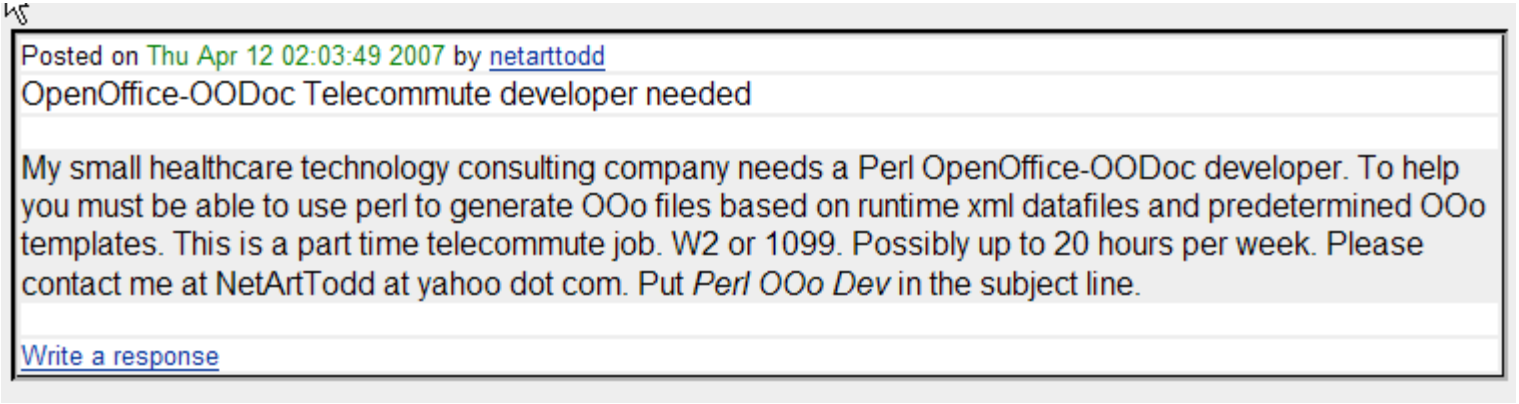
```
////////// Sample Code ///////////////////////
//some text e.g read from a TextBox

string someText         = "Max Mustermann\nMustermann Str. 300\n22222 Hamburg\n\n\n\n"
                        +"Heinz Willi\nDorfstr. 1\n22225 Hamburg\n\n\n\n"
                        +"Offer for 200 Intel Pentium 4 CPU's\n\n\n\n"
                        +"Dear Mr. Willi,\n\n\n\n"
                        +"thank you for your request. We can offer you the 200 Intel "
                        + "Pentium IV 3 Ghz CPU's for a price of 79,80 € per unit."
                        +"This special offer is valid to 31.10.2005. If you accept, we can deliver "
                        + "within 24 hours.\n\n\n\n"
                        +"Best regards \nMax Mustermann";
//Create new TextDocument
TextDocument document                           = new TextDocument();
document.New();
//Use the ParagraphBuilder to split the string into ParagraphCollection
ParagraphCollection pCollection         = ParagraphBuilder.CreateParagraphCollection(
                                                        document,
                                                        someText,
                                                        true,
                                                        ParagraphBuilder.ParagraphSeperator);

//Add the paragraph collection
foreach(Paragraph paragraph in pCollection)
        document.Content.Add(paragraph);
//save
document.SaveTo("Letter.odt");
////////////////////////////////////////////////
```

An Open Document Library – C# Library

OpenOffice::OODoc

The Perl Open OpenDocument Connector.

Relatively complete and established.



Posted on Thu Apr 12 02:03:49 2007 by netarttodd

OpenOffice-OODoc Telecommute developer needed

My small healthcare technology consulting company needs a Perl OpenOffice-OODoc developer. To help you must be able to use perl to generate OOo files based on runtime xml datafiles and predetermined OOo templates. This is a part time telecommute job. W2 or 1099. Possibly up to 20 hours per week. Please contact me at NetArtTodd at yahoo dot com. Put *Perl OOo Dev* in the subject line.

Write a response

Toolkits I've Looked At

Name	Language	WP	SS	Pres	URL
Odfpy	Python	X	X	X	http://opendocumentfellowship.org/projects/odfpy
OooPy	Python				http://ooopy.sourceforge.net/
OpenDocumentPHP	PHP	X	X		http://opendocumentphp.org/
AODL	C#	X	X		http://opendocument4all.com/content/view/13/29/
OpenOffice::OOCBuilder	Perl		X		http://search.cpan.org/dist/OpenOffice-OOBuilder/OOCBuilder.pm
PyOpenOffice	Python	X			http://www.bezirksreiter.de/PyOpenOffice.htm
Odf4j	Java	X	X	X	http://odftoolkit.openoffice.org/source/browse/odftoolkit/odf4j/
OpenOffice::OODOC	Perl	X	X		http://search.cpan.org/dist/OpenOffice-OODoc/

Some demos from J. David

Book available online at:

http://books.evc-cit.info

or in printed form from:

http://www.lulu.com/content/207835

Proceeds to benefit the OpenDocument Fellowship