ResXML Frequently Asked Questions

Paul Hoadley, Logic Squad <`paulh@logicsquad.net`>

$Date: 2004/11/10 13:37:19 $

1. Introduction

1.1. What are the aims of this project?

The overriding aim was to create an XML application for the presentation-oriented markup of resumes or curricula vitae. By ‘presentation-oriented’ I mean that the intention was to make this system suitable for producing actual output from the XML—output that would actually be used in the way resumes are normally used. In other words, the intent was never to make a document type suitable for, say, interchange between corporate human resources departments. The original motivation was drawn from the fact that over a period of years, the nebulous concept that was my own resume had inadvertently spread itself over numerous different **Microsoft Word** files. There were several problems with this. **Word**'s ‘`.doc`’ format is an awful choice from an archival point of view, and possibly an even worse choice from a presentation point of view. It's quite conceivable that within a few years, current versions of **Word** won't even be able to open older versions of my resume. When that time comes, the data contained in those files is essentially lost. The idea of having a central repository for the information itself that is separate from the eventual presentation is appealing, and XML was the obvious choice for achieving this. It simply inverts the old model: now, there's one file with all the information, and I produce versions of the resume tailored to a purpose as I need them—when they are no longer required, they can be deleted safe in the knowledge I can regenerate them if ever required. This contrasts with the historical situation: the data is spread over multiple files and I need to actively retain old resumes as they are the only repository of the information itself.

1.2. Why do we need yet another XML application for marking up resumes?

The basic answer here is that I didn't like what was freely available. A resume or curriculum vitae is a relatively personal document, and I like mine structured in a particular way (in terms of both the information content and the eventual presentation). At the same time, my intention was always to make that structure fairly general so that it might appeal to other people as well. Before starting the project, I surveyed the available XML offerings. Certainly Sean Kelly's work with XML Resume Library is impressive (and some of my own schema is inspired by it) but it just wasn't quite what I wanted. For example, I prefer my achievements and employment record to largely stand on its own, at the expense of more abstract ideas like a section for a person's overall objectives. Your mileage may vary.

There is a group called the HR-XML Consortium who develop an XML application for resume data. The following is how they describe themselves on their website:

> The HR-XML Consortium is an independent, non-profit organization dedicated to the development and promotion of a standard suite of XML specifications to enable e-business and the automation of human resources-related data exchanges.

The ResXML Project has a completely different focus: generating something for a job-seeker to hand over to a prospective employer. I have not had an extensive look at the HR-XML DTD, but our aims would seem sufficiently divergent to assume that there is room for two different models.

2. Using the schema and stylesheets

2.1. What output formats are supported?

Currently there are XSLT stylesheets to produce HTML output and XSLFO for transformation into PostScript or PDF. If you need plain text, get a plain text HTML browser such as Lynx to dump the HTML version to file. If you need **Microsoft Word**'s '`.doc`' format, I'd first re-evaluate whether you really want to work for someone who believes **Word** to be a universal interchange format, and then investigate the Jfor project as a means of transforming XSLFO to RTF.

2.2. What operating systems are supported?

One of the nice features about XML, and most applications based on it,[1] is that it is largely operating system neutral. The software requirements for using this system are considered below, but basically any operating system for which there are a set of appropriate XML tools will be just fine.

2.3. What XML processing tools will I need?

Although you can get by without one, an editor that knows about XML syntax is a great help. Good editors can ensure that you are using only the permitted

[1]Here I am using the term 'application' to mean an XML schema and associated transformation stylesheets rather than a conventional application that just happens to use XML as a data format.

tags at the right time, can prompt you for attribute values and automatically close tags.

To produce output from your resume marked up in XML, you need first an XSLT processor. This tool will transform your XML source into either HTML or an intermediate format called XSLFO on the way to print output. A non-exhaustive list of XSLT processors (and their authors in parentheses) follows:

- xsltproc (Daniel Veillard)
- Saxon (Michael Kay)
- Xalan (The Apache XML Project)

The tools written in Java (Saxon and Xalan) are available for any platform which has a JVM, including **Microsoft Windows**.

To generate PostScript and PDF from XSLFO requires an XSLFO renderer. Some open source implementations exist, though none are complete:

- FOP (The Apache XML Project)
- PassiveTeX (Sebastian Rahtz)

The **ResXML** XSLFO stylesheet output is tested with **FOP**, and the resulting PDF is perfectly acceptable. There are two commercial products available, and, although expensive, they are much closer to full implementations of XSLFO:

- XEP (RenderX)
- XSL Formatter (Antenna House)

Personally, I recommend XEP. It is written in Java, and will run on any operating system that has a JVM (including **FreeBSD**, **NetBSD**, **OpenBSD**, **Linux**, **Solaris** and **Microsoft Windows**).

2.4. So how do I actually make a resume?

There are two basic steps involved: marking up your resume as an XML document conforming to the ResXML schema, and then processing this document into one of the output formats.

The easiest way to mark up your resume is to use an XML editor of some kind. This could be either a dedicated XML editor (for example, xmlspy from Altova), or an ordinary text editor that is XML-aware in some way (such as GNU emacs from the Free Software Foundation and the PSGML or nXML editing mode). You will need to point the editor at the ResXML DTD (for PSGML) or the Relax NG schema (for nXML). How you do this will vary between editors. Using GNU emacs and PSGML, for example, it is sufficient to provide the location of the DTD in the `DOCTYPE` element:

```
<!DOCTYPE resume SYSTEM "resume.dtd">
```

Note

At this point, if you are using GNU emacs and PSGML, press **Ctrl-C-Ctrl-P** to re-parse the doctype and read in the DTD.

The procedure for getting started will vary between editors, though it should be well explained in your editor's documentation. Similarly, the procedure for creating and adding content to elements in the document will vary. Your editor should be able to discern that the top-level element is `resume` from the doctype. From there, you should start by creating a `header` element and providing the information that belongs there.

Once you have marked up your resume as an XML document, you are ready to use the XSL stylesheets to transform it into HTML or PDF. Performing this step is dependent on which XSLT processor and FO renderer you are using. As an example, we will look at how to use **xsltproc** and **XEP** to transform the XML source into PDF. Firstly, transform the XML into XSL-FO using **xsltproc**:

```
> xsltproc -o myresume.fo resume-fo.xsl myresume.xml
```

This will produce an XSL-FO file called `myresume.fo` from the input file `myresume.xml`. You can then render this to PDF with **XEP**:

```
> xep.sh -fo myresume.fo -pdf myresume.pdf
```

Note

This assumes `xep.sh` is in your command path. You can provide an absolute path to `xep.sh` if it is not.

You should now have your resume formatted as a PDF file called `myresume.pdf`.

2.5. I know what I'm doing—where do I get the distribution from?

You can download a compressed **tar** archive of all the source code from ResXML's SourceForge.net File List page.

3. Contact and contributions

3.1. Who are the authors?

- Paul A. Hoadley
- Philip Roberts

3.2. Is there a mailing list?

Yes. There is a list for users and a list for developers, both hosted at SourceForge.

3.3. Can I help out?

Sure. Check out the ResXML Project page at SourceForge. Download the code. Join one or both of the mailing lists. Tell us what you think.