

## Appendix 2

# OPLÉ technical development issues

*The following text is closely based upon a paper presented by Dr Alex Brown and Francis Cave at the XTech 2007 conference in Paris on 16 May 2007. It describes OPLÉ in technical terms.*

## The Need for an Editing Application

Such a complex XML language as ONIX-PL is almost doomed from the outset without adequate tool support. The target audience of the language are not primarily technologists, so it is unrealistic to expect that they should handle XML markup directly.

## A FOSS solution?

A requirement of an editing solution was that it be developed using, and be made available as, free open source software (FOSS). Thus a sourceforge project was created, a BSD license adopted, and the search began for a viable technology platform on which to base an editing solution.

The developers, Alex Brown and Francis Cave, had had experience building web applications before and were apprehensive. Typically the forms-like portions of such applications are tedious to code and maintain. ONIX-PL, with its richly structured documents would clearly be an impossible basis for any kind of hand-crafted approach. The situation was complicated even further that the schema is expected to be in a state of rapid change, both during and after the project!

Even more challenging than these considerations was the fact that editing ONIX-PL instances is not simply a matter of, well, editing ONIX-PL Instances. Instead, the editing process is spread across a more complex workflow – the schema itself acts as a loose kind of ur-grammar, but users are expected to build ad-hoc specialisations of the forms suggested by this schema as a basis, themselves, for creating actual form instances. In reality, then, the kind of documents being

edited are of two main kinds: templates (forms based on the schema which constrain the schema even more), and instances (forms based on templates).

Clearly any solution would have to be very good at allowing XML to be edited to cope with this mixture of challenges.

The project team had already agreed that a client/server application with a thin (web browser only) client was required.

This, combined with the FOSS requirement, led naturally to a choice of J2EE server and the decision to leverage the panoply of tried-and-trusted XML processing software available for Java: Xerces-J, Saxon, etc.

For the development framework the need for a rich user experience initially suggested some of the more recent AJAX-like frameworks such as JSF. However, the particular features of these did nothing to assure us that the underlying problem – of editing vast quantities of XML – would be any more of a pleasant development experience.

This essential XML-centric nature of the project, and our developer preference for standards-based solutions, all pointed to one obvious technology: XForms. The undeveloped state of XForms support in browsers meant we would need a server-side solution, and so the field narrows to one contender: Orbeon Forms ([www.orbeon.com](http://www.orbeon.com)).

We were somewhat acquainted with some Orbeon technology through our standardisation work for ISO. Orbeon uses an XML pipelining language called XPL which we had considered as a candidate for DSDL (<http://dsdl.org/>).

If we had wanted an XML-based, standards-based approach, then Orbeon Forms certainly delivered: XML pipelining, XForms, XHTML-based markup and XPath/XSLT are the centrepieces of the framework's functionality, with other specifications like XQuery, XPointer, and XUpdate coming to the party.

## Development Experiences

Development of the mooted system took place in the last months of 2006 and the first quarter of 2007, using a two-person team.

## A World Without Code?

Sean McGrath (a leading XML expert) has written that the next big programming language won't be a programming language at all. Indeed, one of the most notable features of this project is how little conventional imperative programming was required – the entire OPLE application required that only approx. 60 lines of Java be written; the rest of the application is entirely expressed declaratively using XML.

## Deriving Forms from Schemas

It has long been a sort of holy grail of form creation that they should be derived automatically from some source schema. The unfortunate fact is, however, that in all but the most trivial cases, schemas themselves contain too little semantic information for an automated process to be able to derive a reasonable form interface for them. Some questions are merely stylistic (are text elements single fields or multi-line?), but more fundamental are questions of how to deal with optional and repeatable elements.

The approach taken by OPLE to is to introduce annotation into the schema that acts as hints for a forms engine so that it might produce a more usable form. The annotations exist as attributes in their own OPLE-specific XML Namespace and so may be cleanly separated from the schema itself using XSLT, NVDL or other processing technologies.

## Upsides

- Fit. XForms is just such a perfect fit for applications which perform a lot of XML manipulation. The “bang-per-buck” this brings is a constant pleasure.
- Standards-based. As a developer, one enjoys re-using XML standards knowledge already known, and the worth developing new skills that are likely to be re-usable in future.
- Declarative XML fails safely. When developing, errors that occur are usually from a fairly narrow set of high-level data errors (e.g. When a wrong XPath results in an incorrect value being retrieved, or an XSLT stylesheet produces unwanted output). The world of null pointers and exceptions is far behind.

- Few restarts. An ancillary benefit of there being so much declarative code is that it can be developed in situ in the web application, without the need to frequent server restarts.

## **And the downsides?**

- As with much web application development, debugging can be somewhat tortuous. This is particularly exacerbated using a framework such as Orbeon Forms as the cause of errors can be particularly difficult to find when there are so many levels of indirection in play. Thus an XML parsing error reporting missing whitespace between PUBLIC and SYSTEM identifiers might, in fact, mean that the database store is offline!
- The learning curve was steep, and in the early days seemed near-vertical, even for developers who would count themselves as seasoned XML hackers.
- The open source software itself is something of a moving target. Orbeon Forms itself has proved remarkable stable though one notices a steady sequence of fixes and improvements being effected through 'nightly builds'. The eXist database used for storage proved somewhat more problematic especially before upgrading to a recent release, and still has bugs which we have to work around in code.
- Performance has been a problem: we have had to devise strategies for splitting and re-assembling forms to keep the application responsive. Even so, while the application works well for its envisaged single-user-per-server scenario, one wonders how scalable server-side XForms are.

## **A Vindication of Standards-based FOSS**

As a finished piece of developed software OPLE is something of a vindication of free and open source software. OPLE itself is a complex web application which deals with a number of tedious real world problems.