Converted 2011-09-01 11:04 from the Web using Citavi 3.1. Differences from the original page are possible.

The Linguist's Toolbox and XML Technologies

Chris Hellmuth (Colgate University)
Tom Myers (N-Topus.com)
Alexander Nakhimovsky (Colgate University)

Introduction

The main point of this paper is that the Linguist's Toolbox should be integrated into a larger software framework that we will call The Linguist's Computing Platform. The main goal of such a transition is to enable collaboration and shared use over the network. A related goal is to bring open standards to linguists' work: without open standards, collaboration and shared use are impossible.

The computing platform we propose and demonstrate in this paper consists of these components:

- The Linguist's Toolbox or Shoebox (In the rest of the paper, we say Toolbox to refer to both.)
- The Firefox browser
- An HTML editor, such as NVu
- The OpenOffice suite of applications
- MySQL database management system
- A Web server – Apache Tomcat in our version
- Apache Ant, for running command-line Java applications
- Our own software that connects Toolbox data to the rest of the framework.

There may be variations: some people will prefer Postgress over MySQL or Apache and PHP over Tomcat and Java. The point is to create a framework of mutually supportive software components that has the following features:

- All components are free and most if not all are Open Source.
- The entire framework is cross-platform: Windows, Mac and Linux.
- The framework is internet-ready: different components can be on different machines, but they can also all be on the same machine, providing for a seamless transition from individual work to team work to Internet-wide collaboration and sharing.

Most components of the framework use XML formats and XML technologies for data storage, manipulation and interchange. Converting Toolbox data to XML opens a wide range of possibilities thanks to an abundance of excellent tools for processing data in XML formats. These tools include XML parsers, DOM interfaces for processing XML data, and XSLT (eXtensible Stylesheet Language for Transformations).

The rest of this paper describes several ways of using XML technologies for processing data and metadata created in Toolbox. Our goal is to illustrate possibilities; specific applications can be developed in response to the needs of linguistic practice. The various data peregrinations are described by diagrams with comments.

Toolbox to XML

Toolbox itself provides XML export that converts selected Toolbox data into XML documents in which MDF markers become XML tags. The export mechanism uses the .typ file to create a hierarchy that groups together related elements of data files, such as .tx, .mb, .ge and .ps in an interlinear file. However, Toolbox does not provide an import-from-XML mechanism. We have developed an external parser in Java, the BoxReader, that. reads the configuration files of Toolbox (.typ and others) and uses their information to convert Toolbox data files (dictionaries, wordlists and interlinear) into XHTML. The conversion preserves the input information, so its output, possibly edited, can be converted back into Toolbox files. BoxReader is a SAX parser that is used to convert non-XML data to XML, as described in [XMLP] ch. 4.

XHTML is a dialect of HTML that conforms to XML rules. As HTML, it can be displayed in the browser; as XML, it can be processed by XML tools. The output files of BoxReader represent the fields and records of Toolbox by the generic XHTML span container. Since a span can contain other spans, the hierarchical structure of Toolbox data can be rendered by a tree of spans. Tag information is rendered as the values of the class and title attributes of those generic containers. The initial intent of the class attribute was to serve as input to CSS formatting rules, but it is increasingly pressed into additional semantic service, especially in the so-called XHTML microformats. The output of BoxReader may, in fact, be considered an (as yet undocumented) microformat for Toolbox data. The title attribute always has the same value as the class attribute but is not completely redundant: it make is possible to see the value of the class attribute by holding the mouse over an item.

A sample of the BoxReader output of an interlinear file is shown in Figure 1.

Figure 1.

To insure interoperability, we provide an XSLT transform that converts our XHTML rendering of Toolbox data into the XML format of the Toolbox export. Another program can transform that XML data in a relational database, as explained later in this paper. This is summarized in Diagram 1:

Diagram 1.

Diagram 2 shows how these XML representations are integrated with the rest of the framework. Note that there are two paths to PDF: one directly from XML export via the XSL-FO transformation into "Formatting Objects," the other via import of XHTML into OpenOffice that in turn provides export to PDF.

Diagram 2.

OLAC Metadata

For users of Toolbox, the best place to create OLAC metadata would be within Toolbox itself, as part of the regular workflow. To this end, Joan Spanne of SIL has created a set of MDF markers that encode a subset of the fields of an OLAC record. The same BoxReader that we use to convert Toolbox data to XHTML can also be used to convert OLAC metadata (or any other MDF-marked data). Once so converted, we apply an XSLT stylesheet to it to produce an XML document that holds the OLAC records in the standard format. Another XSLT inserts those records into an OLAC "static" repository. (The word "static" is in quotes because the repository is, in fact, dynamically generated in memory from the current contents of Toolbox files; the user can save it to a disk by using the Save As menu command of the browser.) This second XSLT can integrate OLAC records from several Toolbox projects. Diagram 3 shows the movements of OLAC metadata from Toolbox to the static repository.

Diagram 3.

Relational Databases

As Diagram 2 indicates, we provide several channels for storing Toolbox-created data (including OLAC metadata) in a relational database. Relational databases have several advantages over file systems as data repositories: they are easily accessed over the network; they lock records when they are in use preventing collisions; they have an elaborate system of access control; most importantly, they provide a standard and powerful query language, SQL. SQL, especially in combination with RegularExpression filters, makes very fine-graned searches possible. For instance, one can ask for all lexemes whose part of speech is verb, and whose stem ends and ending begins with a consonant. It is also possible to create groupings of characters ("Variables") on the fly, for the purposes of a specific query.

We provide two paths from Toolbox data to relational database tables: one via the Toolbox XML export, the other via the BoxReader and XTHML. In both cases we provide a number of queries that can be entered from an HTML form, with query results viewed in the browser. Users who are familiar with SQL can construct their own queries. Users who are familiar with Regular Expressions can utilize those in their queries.

OpenOffice

Release 2 of OpenOffice is a major upgrade that establishes it as a computing platform in its own right: one can use it as a suite of office applications that includes an HTML editor, and as a database front end. It natively supports several programming languages, and it keeps all its data in a standard XML format (OpenDocument). The standard has been developed by the Organization for the Advancement of Structured Information Standards (OASIS). A complete RELAX NG grammar for OpenDocument can be found at http://www.oasis-open.org/committees/download.php/12571/OpenDocument-schema-v1.0-os.rng. As of May 6, 2006, the OpenDocument Format (ODF) is also an official ISO standard ISO/IEC 26300.

Importing an XHTML document into OpenOffice is trivial: it can simply be opened in OpenOffice and saved in the OpenDocument format. The path from Toolbox to XHTML to OpenOffice thus offers an alternative to Toolbox export to RTF and MSWord. Just like MSWord, OpenOffice can be use for printing, either directly or after exporting to PDF. OpenOffice has some advantages over MSWord, of which we mention three. First, it is based on open standards while RTF is proprietary and has, in the past, changed in unpredictable ways. Second, OpenOffice supports more scripting languages, including JavaScript. We can write JavaScript code to query and modify the contents of an OO document. (Since it is XML, we can use the familiar DOM interfaces to do so.) Finally, OpenOffice supports direct database access. (Imagine that you can query a SQL Server database from MSWord and integrate the results into your document, all of it for free.)

Note that we now have two database front-ends, the Firefox browser and OpenOffice. They can be used in complementary fashion: OpenOffice by the individual researcher who is creating or editing the materials, for both SELECT and UPDATE queries, and the Firefox browser for read-only access over the internet when materials are shared with other researchers. Exporting Toolbox data from OpenOffice requires an XSLT filter that transforms OpenDocument XML files into the XHTML structured as the output of BoxReader. Since the output of BoxReader can be re-imported into Toolbox, we have, in effect, another WISYWIG editor for Toolbox data. The advantage of that editor is, as before, that it is internet-ready for collaborative work and shared results of that work.

Conclusions

Integrating the Linguist's Toolbox into the distributed Computing Framework opens many new possibilities for working with language data, both individually and collaboratively. We will be preparing a CDROM with the framework software and installation instructions. Our own software will be released under an open source license.

The biggest next challenge is to identify the most important needs and possible scenarios of use within the framework. We are counting on help from practicing linguists in identifying those needs and scenarios of use.

Acknowledgements

Work on this paper was partially supported by the NSF grant #0553546 under the Documenting Endangered Languages program. We are grateful to Joan Spanne of SIL who shared with us her work on integrating OLAC into Toolbox. We would also like to acknowledge help by Alan Buseman, Karen Buseman and Gary Simons, also of SIL; Denis Paperno of Moscow University; and Hannes Hirzel of the University of Zurich.

References

[XMLP] Nakhimovsky, Alexander and Tom Myers. *XML Programming*, Apress, 2003