XSLT: where does it fit in?

Julia Flanders, Syd Bauman
No one needs to convince us of the importance of the overall topic here: "transforming and publishing TEI". It's why we create TEI data. However, we may need to do some preliminary clarification and scoping to get a full sense of what we mean, of what the possibilities are and what kinds of "publishing and transforming" they entail.

One way to orient ourselves in the landscape of "publishing and transforming" is to think about how we use our own data. If we think of the life cycle of a TEI project, there are numerous places along the timeline where we want to express different views of our data, for internal or external viewing:

- proofreading and error catching
XSLT: where does it fit in?

- formats that extract specific structures to let us catch inconsistencies in the encoding
- web publication
- print publication
- formats for contribution to collaborative projects (where we might want to simplify or alter our markup to match the target encoding of those projects)
- metadata formats (e.g. to expose to metadata harvesters)
- archival formats for committing to a repository

Single-source publishing and XSLT
Another way to approach the topic of this workshop is to think about transformation and publishing as a variety of informational avenues that radiate out from our TEI data. Even though it's probably a familiar concept to many of you, it's worth noting an important assumption that underlies much of our work with XML: we're creating a single XML source from which we are going to generate many different kinds of output.

This is important because the XML source is an expensive and valuable information object: it represents a careful modeling of our research materials, we've put a lot of work into it (transcription, encoding, proofreading, correction, annotation, other kinds of enhancement) and we want to exploit it in many different ways, automatically, not by hand.

When we generate these different varieties of output, we are often losing information: erasing distinctions that are present in the source (but unnecessary in the output), or moving from a representationally rich language (like TEI) to a representationally impoverished language (like HTML)

But since these output formats are generated automatically, rather than by hand, this information loss doesn't matter: the source retains its informational richness: it represents the full set of possibilities from which any specific option can be generated.

Some examples

Charles Brockden Brown Archive
Mark Twain Project
Paul the Simple, XML source

A few examples:
• Charles Brockden Brown
• Mark Twain Project
• Paul the Simple
A third important aspect of our topic is the idea of data as a mutable, protean substance: as a kind of plastic informational model that we can reshape and manipulate as needed.

In the example here, all four of these examples represent pretty much the same pieces of data—any one of them could be generated from any of the others. And yet these differences might matter in the context of some particular tool or standard way of doing things.

The point is that our data is almost never trapped in its current format: when we understand it as transformable, we gain power over it and we can use it more flexibly. If a collaborator
needs some information extracted from our data, or if they put their fields in a slightly different order, or whatever, it's not a problem.

Scope and ambition

The chief tool for doing all of these kinds of work is a programming language called XSLT, the Extensible Stylesheet Language for Transformations:

• it can be used on its own to generate different kinds of transformed and manipulated data (such as HTML, KML, JSON, other XML formats)
• and it also is built into many (most? all?) of the XML publication systems that we use, such as XTF, as the way that they take XML data and manipulate it as part of their publication activities
Either way, what it does is give us a way of manipulating our XML data: to extract pieces of it, reshape them, change their format, generally do whatever we want to do with them.

Let's talk for a moment about what we're going to cover in this seminar (and what we're not going to cover).

This seminar is aimed at people who have TEI data and not much else: we aren't assuming familiarity with programming, or with XML publishing tools.

Our goal is to help you learn about what's involved in using your TEI data: in publishing it, in manipulating and transforming it into other formats, exploiting its informational potential; we'd like you to come away, first of all, with a sense of what is possible.

How about in concrete, practical terms? What are we actually going to cover? XSLT is hugely powerful--it is a full-fledged programming language--but as a result it's a big topic:

- hence we are not aiming here to teach you XSLT in any kind of comprehensive way
- what we are aiming to do is give you an understanding of how XSLT works, both on its own and in the context of XML publishing systems
- on the first two days, we are going to look at a lot of examples, and we're going to experiment with a lot of things that XSLT can do, including generating HTML and KML
- on the third day, we are going to install and set up two fairly simple XML publishing tools: XTF and TEI Boilerplate
- so at a minimum, by the end of the workshop you will be able to take your TEI data and publish it on the web in some basic ways.

By the end of the workshop, you should also have a good sense of whether XSLT is something you want to know more about and learn in a more systematic way, and if it is, we encourage you to take a more intensive XSLT workshop: Syd teaches one at DHSI, and Syd and David teach one at Brown every so often. This workshop is a good starting point for either of those workshops.

---

Simple Publication with XSLT

Extensible Stylesheet Language transformations allow you to transform XML documents into other formats
The Extensible Stylesheet Language allows you to transform XML documents into other XML formats. Essentially XSLT allows you to map a given XML element onto another XML element: saying "take in the following construct, and put out this other construct." It could be a construct in the same language, or in a different language such as XHTML, as in the example here.

XML Databases and Publication Frameworks

Tools designed to manage large groups of XML files, with more advanced functionality:
- fast, efficient searching
- transformations involving groups of files
- eXist, DBXML, Xindice, XTF, MarkLogic
XSLT: where does it fit in?

The XML database and publication framework universe

These kinds of tools are designed to manage large groups of XML files, and to provide certain kinds of advanced functionality:

- fast, efficient searching
- transformations involving groups of files: not just transforming each file separately, but doing transformations that involve taking parts of different files and creating new results files: for instance, a sorted list of the first lines from all the poems in a collection.

How do databases fit into a larger XML publication framework? What do they do?

- they create and store indexed information: that is, information from the source XML files that has been preprocessed to make it more accessible and easier to manipulate. For instance, they might store tables of all the document metadata (author, title, genre, date, etc.) so that it can be searched and sorted more quickly.
XSLT: where does it fit in?

• they contain a representation of the document's structure in a format that makes it easier to process, so that certain kinds of navigation are easier
Within the XML publication framework, the database sits and waits for queries to come in. • when it receives a query, it performs the necessary searching and returns a result (in the form of an XML fragment, or a node set, or some proprietary structure)
• the result can then be transformed (e.g. into HTML for delivery to a browser, or into some other XML format for other processing) using XSLT

XML databases exist as separate modules that can be used as the basis for XML publishing systems, for instance:
• eXist
• DBXML
• Xindice (Apache)

The Bigger Picture

What you want to do

Basic document display, navigation  Searching, visualization  Data mining  Production-level publication, access control  Long-term sustainable publication, data integrity over time

Small data set  Large data set  Multiple data sets

Small, Simple, Cheap  Big, Complicated, Expensive

XSLT, CSS  XML database  Large-scale publication frameworks

Desktop + you  Web server + simple technical support  Web server + professional systems administration  Server + Institutional IT  Institutional repository + data curation

What and whom you need

The tools you need, and the people you need, can be imagined as a rough continuum of increasing scale, complexity, difficulty, and cost:
at the simplest level, there are things you can do (or learn to do) by yourself, with very little in the way of equipment or software: tools like XSLT and CSS will go a long way towards producing simple, effective interfaces for browsing and reading small sets of documents

at a slightly more complex level, as the number of documents increases and as you want to do more ambitious things with them (such as visualizations, complex searching), you need software tools that are a little more challenging to manage: perfectly within the capabilities of a humanist, but requiring more time: not something you can do on the side of another job; this becomes someone's major job responsibility

Going a bit further, we get to things that require XML publication frameworks that require a professional systems administrator, someone who really understands the installation and configuration of things like web servers, XML databases, etc. These are the kinds of tools we need to build things like data mining or text/topic analysis into our publications, and also if we want to publish larger collections of documents that require more server power/speed

For production-level publication, where you may be actually charging money for access (and hence need to do things like authentication) and hence may have higher standards of performance and reliability, you need to start engaging with your institutional IT organization to make sure that things like backups, server maintenance, etc. are being handled at the appropriate level of professionalism; this is also the level of scale at which we start to be able to really work effectively with multiple large data sets: for instance, multiple projects of substantial size

Finally, if we want to be able to ensure the long-term sustainability of projects, we need to engage with systems like institutional repositories and the data curators who can help us ensure that data will be maintained, migrated, etc. after the project itself is no longer funded.

So considering where the three examples we looked at earlier might fit in:

- Paul the Simple: a single scholar, acting alone
- Charles Brockden Brown: a small amount of professional systems adminstration
- Mark Twain Project: a much larger staff, embedded in the CDL (which is where XTF was built)