SBCT Web Programs: Project Documentation

https://rooneypower.github.io/sbct/

Overview and Significance

The South Bend Civic Theatre (SBCT) is a leading community performing arts organization in South Bend, Indiana with over 50 years of performance history. Many of the performers have ties to other organizations in the area, particularly educational institutions such as the University of Notre Dame. My family has attended many SBCT performances since coming to South Bend in the 1980s, and we have occasionally contributed our amateur talents on the stage.

For families such as mine, theatrical programs from SBCT represent the history of our community and some of the better memories woven into the fabric of our lives. This project undertakes to publish a selection of those programs on the open web, making such memories available to anyone with whom they might resonate. The goal is to highlight the people at the heart of the productions in a simple, clean format. Behind the HTML display, the basic structure of the programs is encoded in XML. This represents a first step toward enabling the automated extraction of individual roles, names, or other data that may be of scholarly interest in the future.

Source Material

The SBCT Digital Program Library is a collection of 25 theatrical programs from the South Bend Civic Theatre. It is hosed on Omeka.net. Each program is presented as a scanned PDF with accompanying metadata. The full program text, excluding certain advertisements and other unrecognized material, is included in the metadata to facilitate full-text search. It was digitized using optical character recognition (OCR) and limited manual corrections.

Omeka can output item data in multiple formats. This project uses omeka-xml as the input for its transformations. While the schema link for omeka-xml returns a 404 error and detailed documentation does not appear to be readily available online, all the necessary content is accessible in well-formed XML. Most significantly, the full program text for each item can be found at //item/itemType//element[@elementId='1']//text. The text itself is only structured with line breaks (encoded as &#13;) and semi-predictable section headings (e.g., the crew may be labeled as "The Production Team," "Production Team," or "The Production Crew"). Omeka-xml also includes Dublin Core (DC) metadata elements such as Title and Creator, encoded using Omeka element IDs.

The actual content of the programs varies based on when they were created and the type of production, but they all feature a cover, some sort of cast and crew listings, and biographies for the talent involved. For this project, human information is of primary interest and that is where extraction efforts are focused. Several of the programs are for musical performances and feature extensive song listings. That information is irregularly formatted and may not appear in the digital publications.

Data Format

In order to publish the programs in an appealing web format, they first had to be converted to something more structured than text with line breaks. Accordingly, I devised a simple XML format specifically for this purpose. Each XML document represents one program, so the root node is <program>. It has two children: <metadata> and <text>. The metadata element contains basic DC metadata extracted from the omeka-xml, specifically <dc:title> and <dc:creator>. The text element contains up to five children, corresponding to key sections of the program: <cover>, <cast>, <crew>, <setting>, and <bios> (biographies). Each section can contain one <header> and an unlimited number of content elements. The content elements vary by section. For <cover> and <setting>, the content element types are <line> and <break> (for lines of text and blank separator lines, respectively). For <bios>, each short biography is an <entry>.

The cast and crew elements have the most defined structure. Each line from the list in the original program becomes a <credit>, which contains a <role> and a <talent>. The role element contains the theatrical role or production position title, and the talent element contains the name of the person filling that role. If multiple people are listed for a position, they typically appear together in a single talent element (delimitation in the source is sufficiently irregular to make more granular parsing difficult). This nested structure is designed to be readily displayed in tabular format.

Publishing System

The publication process consists of four main steps:

1) Download the item data from Omeka.net in omeka-xml format.
2) Unpack and transform each item into its own file, marked up in this project's original XML format for theatrical programs. This step is handled via a two-stage XSLT transformation.
3) Define how the programs will appear on the web. This is accomplished by a second XSLT transformation, applied as a stylesheet via processing instruction.
4) Display the documents. The project is hosted on GitHub Pages. Each program appears as an HTML webpage when viewed in a browser that executes the XSLT stylesheet.

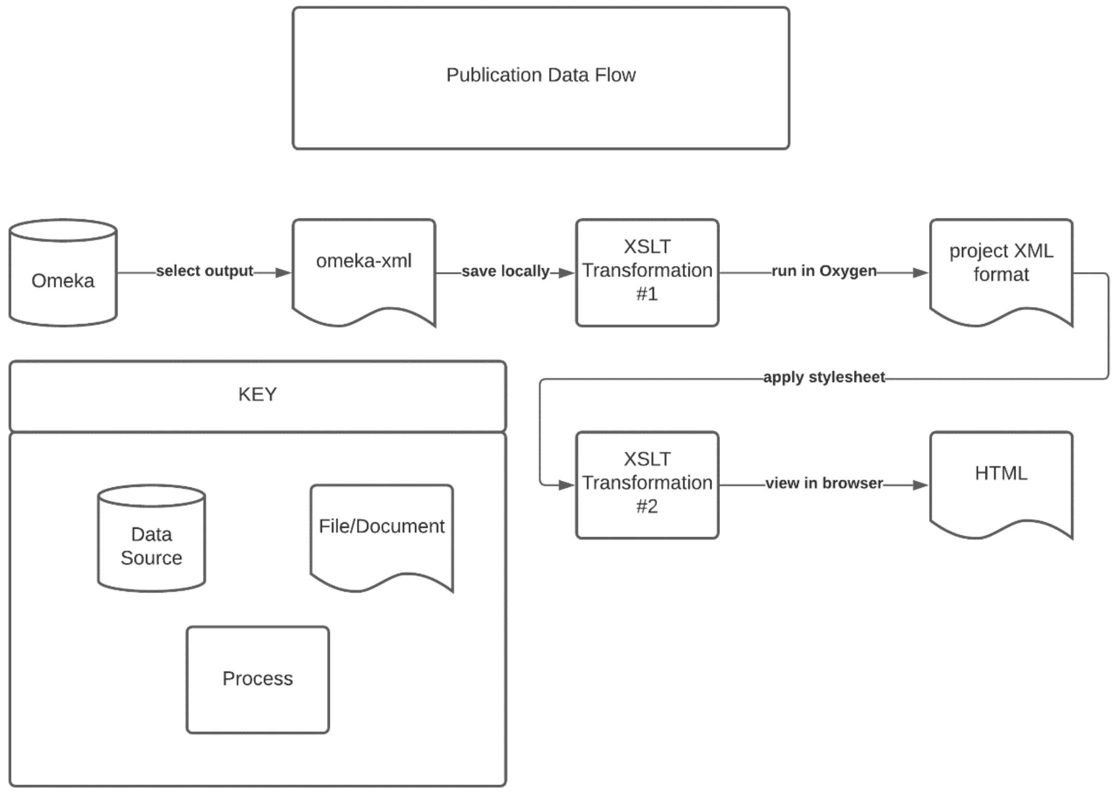
Step One is easily accomplished via the administrative interface on Omeka.net. Batches of items can be selected simultaneously, and clicking the appropriate link outputs them together in a single XML document. The 25 items for this project were downloaded in three batches (10, 10, and 5). Each file has an itemContainer root element which wraps the collection of item elements.

Step Two is the most technically challenging portion of the project. Breaking each input file into individual output files for each program is simple enough using <xsl:result-document>, and the DC metadata can be extracted with standard XSLT templates. The complexity comes in handling the actual text of the program, which is stored in a single text element within each program's metadata. Other than breaks between lines, there are few reliable delimiters in the text. Not all programs contain the same sections in the same order, or use the exact same strings to indicate the beginning of a section.

The transformation I developed for this purpose, program_parser_pi, makes two passes through the document. The first recursively creates <line> and <break> elements from the plain source text; the second attempts to build those lines into meaningful data structures based on position relative to recognized strings. The goal is to extract the key information in a standardized format, rather than to recreate all the content and styling of the original documents. The transformation is run in Oxygen XML Editor using Saxon-PE 9.8.0.12. Each output file contains one program in the new XML format developed for this project, with a processing instruction defining the display format via another transformation.

That second XSLT transformation, program_style_pi, is the key to *Step Three* of the publication process. It functions as a stylesheet, transforming the XML into HTML styled with CSS. Since the difficult processing is handled by the previous transformation, this one consists largely of simple templates mapping individual element types to their appropriate HTML counterpart. The program title is used as the page title and followed by a link to the corresponding item in Omeka. The cover and setting sections consist of short, centered paragraphs (similar to the original paper programs). The cast and crew sections are displayed as side-by-side tables. The biographies appear as left-aligned paragraphs, with performer names emphasized when readily identifiable.

In *Step Four*, the XML documents from the second step are uploaded to GitHub. When viewed in a browser, the files are automatically displayed as HTML pages per the stylesheet transformation outlined in step three. Those pages are organized into a simple website using GitHub Pages Jekyll-based framework.


Publication Data Flow
Omeka
select output
omeka-xml
save locally
XSLT Transformation #1
run in Oxygen
project XML format
apply stylesheet
XSLT Transformation #2
view in browser
HTML
KEY
Data Source
File/Document
Process