Appendix: Database Software, Scripting, and XML

Chapter two provided some of the basics of creating a website, but those contemplating more complex sites need to think about more complicated technical infrastructures, especially the possibility of organizing expansive resources through databases or XML. If you plan on having more than a hundred digital objects in a historical archive, collecting more than a hundred historical artifacts or documents through your website, or are responsible for the membership roll of a historical society, you should consider attaching a web-enabled database to your site. If you are digitizing a similarly large number of historical documents that are mostly or completely text, you should consider the possibility of marking up those documents with XML. Using either a database or XML generally adds a few layers of technology to a more basic website: additional software to encode or store materials; a way to deposit information into a database or markup texts with XML; and a way to extract information from the database or XML archive to be displayed on a web page.

Databases

Most historians encounter simple databases when they use common software applications like Access or Filemaker. Web databases are different than these “client,” or desktop, programs. They run silently in the “background” of a web server and respond to specialized requests through small pieces of programming code rather than through the point-and-click actions that occur on a personal computer screen. These invisible database operators are fast and allow for complicated “queries,” or instructions, from a web page.
Your budget and the other technologies you plan to use on your site will help you choose an appropriate database program. If you anticipate having more than 100,000 entries in the database and require no corruption or loss of information and a service-oriented company (in other words, if you are in charge of a large projects like Ellis Island’s website, which stores millions of names as well as associated documents), then you may wish to use software from Oracle, the high-end market leader, or similar products like DB2 from IBM. These expensive database products are often sold with technical support and cost thousands of dollars per year or more to buy and run.\(^1\) Even then, many large historical sites with some in-house technical capabilities may be able to use a free high-end alternative called PostgreSQL, which has most of the same features as the commercial software with none of the eye-popping cost.\(^2\) For example, the Eastern Illinois University students and professors behind the Coles County Legal History Project, which is cataloguing and making searchable via the web legal documents from the era of Abraham Lincoln, Esq., decided to move from Access to PostgreSQL as their collection and website grew rapidly over the years.\(^3\) PostgreSQL, like Oracle’s products, is “ACID” compliant, an acronym for a database checklist that ensures data integrity during fast-paced, high-volume usage.

For all but the most extensive historical archives, however, PostgreSQL, Oracle, DB2, and other robust databases are overkill, and many good alternatives for small- to medium-sized historical websites exist. Microsoft SQL Server sometimes comes bundled with Internet Information Services (IIS), Microsoft’s web server software (beginning at around $1,500), and in certain ($20,000 and up) versions it can handle as much information as any database (educational versions of SQL Server are considerably less). In addition, SQL Server includes administrative software that maintains Windows’s look and feel and thus may feel more comfortable than other
database packages (though you may not see this administrative software if you do not own or run the server it’s on).

Microsoft’s low-end personal database, Access ($229 alone, though also available as part of the Office suite of programs), meant to run more on client computers than web servers, is inexpensive and can be pressed into service as a web database, though not without some software linkages that will require some technical knowledge. The University of Minnesota’s Immigration History Research Center uses Access to store information about its thousands of documents and images, though they convert the Access data into XML to make it searchable via the web.4

More recently, with the widespread adoption of open source software, a good free database alternative to all of the above options has emerged: MySQL, the leading free database.5 MySQL runs on virtually any type of server and ably handles ten or even hundreds of thousands of documents, as it does for our September 11 Digital Archive.6 Although perhaps not as robust as PostgreSQL or Oracle, MySQL is extremely capable for most of the tasks historians will ask of it (e.g., finding a specific document) and is slowly gaining many of the high-end features and stability of its rivals. MySQL will likely continue to proliferate given its undeniably attractive price. Most commercial web hosts (see below) provide MySQL for their customers who wish to have a database attached to their site.

But how do you access the database software lurking on the server, either to put materials in or to get things out to display on your website? In general, putting things in is easier since it can be done through various interfaces without programming. Many database programs come with, or allow for, web-based interfaces to enter data, though you may find such interfaces lacking many of the features you are used to with programs running on your personal computer (like Access, Excel, and Filemaker). The advantage of such web-based interfaces, of course, is
that you can gain entry to them from any computer with an Internet connection and a browser, which means that data entry can be distributed easily. Another possibility is using special linking software to open a web server database within Access or Filemaker on your desktop. The Open DataBase Connectivity protocol (ODBC) enables different client and server database software to function seamlessly together in this way. If you are planning on entering your data only once, it may be easier, however, just to record it as you would in one of the easy-to-use client programs (like Access) and then hand off a tab-delimited file (a text file with tabs separating each piece of information, which most databases like Access can easily create) to your web server administrator for ingest into a web database.

Unfortunately, getting information into the database is only half the battle to using a database on a historical website. You now need to be able to extract bits of this information, format them, and insert them into web pages as visitors to your site make requests. The database software is merely interested in storing and serving this data and cannot do this task for you. Neither can HTML, which is a fairly “dumb” set of text tags. Instead, you must rely on one of several “scripting” languages that know how to communicate with the database software.

“Scripts,” or little programs that you can place either inside the HTML pages you construct or in a separate file linked to a web page, are processed on the fly by the web server and can fill a page with information from a database. There are as many scripting languages as databases, though you often see certain languages in combination with certain databases. For instance, one of Microsoft’s scripting languages, ASP, is generally used with other Microsoft products, including SQL Server. PHP, the open source community’s answer to ASP, is most often used with MySQL and PostgreSQL. In contrast, ColdFusion, a popular scripting system from Macromedia, can be found attached to a wide variety of databases.
The words “scripts” and “scripting” sound very humanities-like, but these languages are, for all intents and purposes, programming languages, meaning that they are more difficult to pick up than HTML and far less forgiving to human error. If you need scripts to access a database, you should consider hiring a programmer who knows one of these languages well. Go about designing and building your site without the materials from the database—leaving room in your design “templates” for those items—and then bring in the programmer for a day or two to write pieces of technical code that will add elements from the database. A programmer who knows how to structure a database properly for your collection of historical items in addition to writing the scripts to place those items into your templates should cost around $50-100 per hour (generally a minimum of $1,000 for a basic setup; complex websites like EllisIsland.org require weeks if not months of database and programming work).

Regardless of who does the technical work, you should start by outlining the specific “tables” for the information you wish to store. Each table holds a specific kind of information—for instance, a list of a historical organization’s members or the pieces of information about letters in an archive. Even without fully understanding database software, you can probably draft the fields of each table: last name, first name, membership renewal date, etc., for a membership roll; author, date written, to whom the letter was sent, body of the text, etc., for a letter. Once in the database, each of these fields will be searchable individually, and in tandem with other fields. From this list of fields (or “columns”) you or your programmer can proceed to set up the framework of the database, into which specific records (also called “rows”) will go. Remember that most of your time (or your staff’s time) will be taken in filling the database up rather than creating it in the first place, but you should start by thinking about which fields to separate out; you will have trouble if you decide later that the last name should be a separate field from the
Although institutional or ISP web hosts rarely provide access to database systems or allow web pages to include programming languages in addition to basic HTML, an increasing number of commercial web hosts provide such amenities. As of this writing, the most popular scripting language among web hosts is PHP, and the most popular database is MySQL. (Note the trend toward free and open source products; you will have more trouble finding a commercial host that uses commercial scripting languages like ASP or ColdFusion, or commercial databases like Microsoft SQL Server.) Most web host comparison sites allow you to list just the companies that offer these more advanced services. If you have a small database or need modest scripting on your site, a commercial host can be a good choice, and they generally charge only a little bit more than for basic accounts for pure HTML sites. Decent web hosting with PHP and MySQL starts at around $10 per month with a single database. Many of these plans also offer access to bulletin board services for visitors to leave comments and engage in discussion about your site or the topic it covers.

**XML**

XML is a more recent technology with less of a track record than the database, but with a growing following in the digital humanities and among librarians and archivists who believe that it will stand the test of time—unlike database files. XML does not have the extensive built-in search features that database programs do, though with some additional fuss you can search XML documents in complex ways. XML is probably best for a historical website with a circumscribed, unchanging set of historical documents that are mostly text. For instance, the Virginia Center for Digital History used XML for their collection of primary documents
(including moving “runaway advertisements”) in their project The Geography of Slavery. XML is also a good choice for archives that wish to share their contents with other related collections since the simple text format XML is written in is, like HTML, rapidly becoming a lingua franca on the web. For example, the Cornell University Library, the University of Michigan Library, and the State and University Library Göttingen combined their historical collections of mathematical works using XML.

Getting started with XML is in some respects much easier— and in others much harder— than using a database. On the one hand, you can use a rudimentary text editor like Notepad to create XML documents (though more sophisticated text editors are very helpful), since XML is simply text with added tags. On the other hand, compared to databases working with XML is a highly unstructured affair. Indeed, you create much of that structure yourself, which is both its beauty and its peril. Although you need to define columns in database programs, many common types of definitions often come preset—for example, database programs have built-in formats for dates and times, and ways to generate such information automatically. By contrast, in XML you have to define each of these elements yourself, although occasionally you can borrow a set of definitions, as with TEI (see chapter three). XML’s flexibility—its ability to tag any bit of text in a document any way you like, highlighting words or phrases as you would with a set of differently colored highlighters on paper—can easily breed unwieldy complexity if you are not careful.

Just as databases require scripting languages like ASP and PHP to pluck information from the database and place it into a web page, XML documents require translators to convert them into HTML for web viewing. Although the very same scripting languages can take care of this task of “parsing” the XML into its constituent parts and putting those parts into a web
template, the World Wide Web Consortium has two technologies specially designed for this task: XSL and XSLT. The Extensible Stylesheet Language (XSL) provides a set of codes for formatting XML elements; the related Extensible Stylesheet Language Transformations (XSLT) converts an XML document with an associated XSL stylesheetsheet into an HTML file that can be viewed in any web browser. With XML/XSL/XSLT (a confusing alphabet soup, to be sure), you can create a formatting template for your XML documents—say, boldfacing the names of authors within each document, taking the date of each letter and right-justifying it—and the server will convert each XML document on request into that format for your web visitors. No need to create separate HTML pages for your web archive; merely create your XML documents and a translator will take care of the rest.

Compared to programming languages like ASP or PHP, XSL and XSLT are fairly straightforward text formats like XML and HTML—that is, they are written in plain English, with few embellishments other than colons and curly brackets. Their syntaxes still require an extra effort to learn, and that effort should not be underestimated, but XSL and XSLT do not require mastering sometimes complex mathematical elements like arrays, functions, and logic in addition to linguistic constructions. Nevertheless, if your head is spinning even slightly from this brief discussion of XML/XSL/XSLT, you will likely have to outsource the creation of these more complex documents and translators. Unfortunately, because databases have been around for so much longer than XML and especially XSLT, many more programmers know how to create websites with a database and a scripting language than with these newer technologies. Furthermore, many more prepackaged (and often free) web tools using databases rather than XML currently exist. For instance, almost all discussion software—which could use either XML or databases—is written for databases. Tens of thousands of programmers who know how to
create websites using the free database software MySQL and the programming language PHP are available; far fewer know XML and XSLT well. This situation will likely change in the coming years, and XML’s large following in the digital humanities provides a source from which to draw strength—and hopefully some advice about implementing this promising technology.

4 University of Minnesota, Immigration History Research Center, http://www.ihrc.umn.edu/.
8 Cohen, history and the second decade of the web