Digital History: A Guide to Gathering, Preserving, and Presenting the Past on the Web

Appendix: Database Software, Scripting Languages, and XML

Introduction

hapter 2 outlined some of the basics of creating a website, but those contemplating more complex sites need to think about more involved 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 If you are digitizing a similarly large number of historical documents that are mostly or completely text, you should consider the possibility of marking up those documents with XML.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 more basic website elements: additional software to encode or store materials, a way to deposit information into a database or mark up texts with XML, and a way to extract information from the database or XML archive to be displayed on a web page.

Digital History: A Guide to Gathering, Preserving, and Presenting the Past on the Web

Appendix: Database Software, Scripting Languages, and XML

Databases

Most historians encounter simple databases when they use common software applications such as Access or Filemaker. Web databases differ from 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 permit quick and 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 to go to with problems (in other words, if you are in charge of a large project like Ellis Island's website, which stores millions of names as well as associated documents), you may want 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 web searchable 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 cost considerably less). In addition, SQL Server includes administrative software that maintains the look and feel of Windows 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 is on).

Microsoft's low-end personal database, Access ($229 alone, and 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, but 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, although they convert the Access data into another format to make it searchable via the web.[4]

More recently, with the widespread adoption of open source software, a good alternative to all of the

preceding 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 quickly) 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 and large base of users ready to help out others. Most commercial web hosts (see below) provide MySQL for customers who want to attach a database to their site.

But how do you access the database software lurking on the server, either to put materials in or to get them out to display on your website? In general, putting things in is easier because it can be done through various interfaces without programming. Many database programs come with, or allow for, web-based interfaces to enter data—for instance, the popular phpMyAdmin for MySQL—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). But such web-based interfaces allow entry from any computer with an Internet connection and a browser and hence facilitate the distribution of data entry. Another possibility is using special linking software to open a web server database on your personal computer within Access, Excel, or FileMaker. The Open Database Connectivity protocol (ODBC) enables different client and server database software to function seamlessly together in this way. If you are planning to enter 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.

[1] The Statue of Liberty—Ellis Island Foundation, *American Family Immigration History Center*, ↪ link A.1.

[2] *PostgreSQL*, ↪ link A.2.

[3] Localités/Localities, *Coles County Legal History Project*, ↪ link A.3.

[4] University of Minnesota, *Immigration History Research Center*, ↪ link A.4.

[5] *MySQL*, ↪ link A.5.

[6] Center for History and New Media and American Social History Project, *The September 11 Digital Archive*, ↪ link A.6.

Digital History: A Guide to Gathering, Preserving, and Presenting the Past on the Web

Appendix: Database Software, Scripting Languages, and XML

Scripting Languages

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 separate files linked to web pages, 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, Active Server Pages (ASP), is generally used with other Microsoft products, including SQL Server. PHP (a self-referential acronym for PHP: Hypertext Preprocessor), the open source community's answer to ASP, is most often used with the open source MySQL and PostgreSQL databases. 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 a programmer for a day or two to write pieces of technical code that will add the database elements. A programmer who knows how to structure a database properly for your collection of historical items and how to write the scripts to place those items into your design 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 want to store. Each table, similar to a spreadsheet in Excel, holds a particular kind of information—;for instance, a list of a historical organization's members or the pieces of data 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, and so on, for a membership roll; author, date written, to whom the letter was sent, body of the text, and so on, for a letter. Once in the database, each of these fields will be searchable individually as well as 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 you (or your staff) will spend most of your time filling the database 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 members' last names should be in a separate field from their first names.

To be more specific, a historical society's membership roll in a database might look as follows:

first_name	last_name	member_since	dues
Charles	Beard	1986	$75
Frederick	Turner	1997	$75
Edward	Gibbon	1999	$75

Programming code, in this case PHP embedded in regular HTML, pulls entries out of the database and formats them for a "membership roll" web page:

```
<html>

<head>

<title>Membership Roll</title>

</head>

<body>

<h1>Membership Roll</h1>

. . .

<?php

// print out today's date

echo 'Generated on ' . date ("F jS, Y") . '<br><br>';

// pull membership information out of the database

$result = mysql_query ("SELECT first_name, last_name, member_since FROM
membership ORDER BY last_name");

//format each member's information

while ($row = mysql_fetch_assoc ($result)) {

echo $row['first_name'] . ' ' . $row['last_name']. ' (member since ' . $row['member_since']
. ')<br>';

}

?>

</body>

</html>
```

The web server processes the programming code when a user requests the page and creates a pure HTML document in a split second as follows:

```
<html>

<head>

<title>Membership Roll</title>

</head>

<body>

<h1>Membership Roll</h1>

Generated on July 4th, 2005<br><br>

Charles Beard (member since 1986)<br>

Edward Gibbon (member since 1999)<br>

Frederick Turner (member since 1997)<br>

</body>

</html>
```

And, unaware of the process that went into its rapid assembly, users see this in their web browser:

Membership Roll

Generated on July 4th, 2005

Charles Beard (member since 1986)

Edward Gibbon (member since 1999)

Frederick Turner (member since 1997)

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 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.[7] 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 for these services than for basic HTML-only accounts. 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.

[7] For example, see *Web Host Directory*, ↪ link A.7.

Digital History: A Guide to Gathering, Preserving, and Presenting the Past on the Web

Appendix: Database Software, Scripting Languages, and XML

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 poignant "runaway advertisements") in their project The Geography of Slavery.[8] XML is also a good choice for archives that want to share their contents with other related collections because the simple text format XML is written in is, like HTML, rapidly becoming a lingua franca on the web.[9] For example, the Cornell University Library, the University of Michigan Library, and the State and University Library of Göttingen combined their historical collections of mathematical works using XML.[10]

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) because 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 3). 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 stylesheet 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.

For example, a version of Frederick Jackson Turner's *The Frontier in American History* prepared in XML and made available for the web, might have the following pieces. First, like any XML document, it must have a Document Type Definition (DTD), in this case defining different parts of the text such as the overall title to the work, the chapter titles, and paragraphs and notes:

<!ELEMENT doc (title, chapter*)>

<!ELEMENT chapter (title, (para|note)*)>

<!ELEMENT title (#PCDATA)*>

<!ELEMENT para (#PCDATA)*>

<!ELEMENT notenumber (#PCDATA)*>

<!ELEMENT note (#PCDATA)*>

<end box>

XML tags are added to the raw text of Turner's *Frontier*:

<!DOCTYPE doc SYSTEM "doc.dtd">

<doc>

<title>The Frontier in American History</title>

<chapter>

<title>Chapter 2:The First Official Frontier of the Massachusetts Bay</title>

<para>In "The Significance of the Frontier in American History," I took for my text the following announcement of the Superintendent of the Census of 1890: "Up to and including 1880 the country had a frontier of settlement but at present the unsettled areas has been so broken into by isolated bodies of settlement that there can hardly be said to be a frontier line. In the discussion of its extent, the westward movement, etc., it cannot therefore any longer have a place in the census reports." Two centuries prior to this announcement, in 1690, a committee of the General Court of Massachusetts recommended the Court to order what shall be the frontier and to maintain a committee to settle garrisons on the frontier with forty soldiers to each frontier town as a main guard.<notenumber>1</notenumber> In the two hundred years between this official attempt to locate the Massachusetts frontier line, and the official announcement of the ending of the national frontier line, westward expansion was the most important single process in American history.</para>

<note>1. Massachusetts Archives, xxxvi, p. 150.</note>

</chapter>

</doc>

<end box>

An XSL stylesheet specifies how to take each part of this XML document and, by mixing its pieces (as

defined in the DTD) with HTML tags, turn them into a new document (technically, an XHTML document) that a web browser can understand:

```
<xsl:stylesheet version="1.0"

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

xmlns="http://www.w3.org/TR/xhtml1/strict">

<xsl:strip-space elements="doc chapter "/>

<xsl:output

method="xml"

indent="yes"

encoding="iso-8859-1"

/>

<xsl:template match="doc">

<html>

<head>

<title>

<xsl:value-of select="title"/>

</title>

</head>

<body>

<xsl:apply-templates/>

</body>

</html>

</xsl:template>

<xsl:template match="doc/title">

<h1>

<xsl:apply-templates/>

</h1>
```

```
</xsl:template>
<xsl:template match="chapter/title">
<h2>
<xsl:apply-templates/>
</h2>
</xsl:template>
<xsl:template match="para">
<p>
<xsl:apply-templates/>
</p>
</xsl:template>
<xsl:template match="notenumber">
<sup>
<xsl:apply-templates/>
</sup>
</xsl:template>
<xsl:template match="note">
<p class="note">
<xsl:apply-templates/>
</p>
</xsl:template>
</xsl:stylesheet>
```

<end box>

The resulting document looks like this:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<html xmlns="http://www.w3.org/TR/xhtml1/strict">
<head>
```

<title>The Frontier in American History</title>

</head>

<body>

<h1>The Frontier in American History</h1>

<h2>Chapter 2: The First Official Frontier of the Massachusetts Bay</h2>

In “T>he Significance of the Frontier in American History,” I took for my text the following announcement of the Superintendent of the Census of 1890: “Up to and including 1880 the country had a frontier of settlement but at present the unsettled areas has been so broken into by isolated bodies of settlement that there can hardly be said to be a frontier line. In the discussion of its extent, the westward movement, etc., it cannot therefore any longer have a place in the census reports.” Two centuries prior to this announcement, in 1690, a committee of the General Court of Massachusetts recommended the Court to order what shall be the frontier and to maintain a committee to settle garrisons on the frontier with forty soldiers to each frontier town as a main guard.¹ In the two hundred years between this official attempt to locate the Massachusetts frontier line, and the official announcement of the ending of the national frontier line, westward expansion was the most important single process in American history.

1. Massachusetts Archives, xxxvi, p. 150.

</body>

</html>

<end box>

In a web browser, this XHTML document would render roughly like this:

The Frontier in American History

Chapter 2: The First Official Frontier of the Massachusetts Bay

In “The Significance of the Frontier in American History,” I took for my text the following announcement of the Superintendent of the Census of 1890: “Up to and including 1880 the country had a frontier of settlement but at present the unsettled areas has been so broken into by isolated bodies of settlement that there can hardly be said to be a frontier line. In the discussion of its extent, the westward movement, etc., it cannot therefore any longer have a place in the census reports.” Two centuries prior to this announcement, in 1690, a committee of the General Court of Massachusetts recommended the Court to order what shall be the frontier and to maintain a committee to settle garrisons on the frontier with forty soldiers to each frontier town as a main guard.[1] In the two hundred years between this official attempt to locate the Massachusetts frontier line, and the official announcement of the ending of the national frontier line, westward expansion was the most important single process in American history.

[1] Massachusetts Archives, xxxvi, p. 150.

As you can see, 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 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 that are found in web programming languages like ASP and PHP, such as 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 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 forum software—which could use either XML or databases—is written for the latter. 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, we hope, some advice about implementing this promising technology.

- Previous

[8] Tom Costa, *The Geography of Slavery in Virginia: Virginia Runaways*, ↪ link A.8.

[9] Daniel J. Cohen, "History and the Second Decade of the Web," *Rethinking History* 8 (June 2004): 297-98, ↪ link A.9.

[10] Cornell University Library, *Distributed Digital Library of Mathematical Monographs*, ↪ link A.10.