Mitigating Madness: How We Authenticate and Authorize Users to Deliver Databases in a Contractually Complicated Context

Jeremy M. Brown
Mercer University Libraries, brown_jm@mercer.edu

Geoffrey P. Timms
College of Charleston

Follow this and additional works at: <https://docs.lib.purdue.edu/charleston>

Part of the [Collection Development and Management Commons](#)

An indexed, print copy of the Proceedings is also available for purchase at:

<http://www.thepress.purdue.edu/series/charleston>.

You may also be interested in the new series, Charleston Insights in Library, Archival, and Information Sciences. Find out more at: <http://www.thepress.purdue.edu/series/charleston-insights-library-archival-and-information-sciences>.

Jeremy M. Brown and Geoffrey P. Timms, "Mitigating Madness: How We Authenticate and Authorize Users to Deliver Databases in a Contractually Complicated Context" (2015). *Proceedings of the Charleston Library Conference*.
<http://dx.doi.org/10.5703/1288284316326>

Mitigating Madness: How We Authenticate and Authorize Users to Deliver Databases in a Contractually Complicated Context

Jeremy M. Brown, Associate Director for Technical Services and Systems, Mercer University Libraries

Geoffrey P. Timms, Marine Resources Librarian, College of Charleston

Abstract

During the 2011 overhaul of the Mercer University Libraries website we developed an authentication system to interface with EZProxy and our campus Active Directory system that provides convenient management and delivery of our A–Z database listing. With multiple campuses and seven e-resource privilege groups, we were able to provide persistent URLs for databases to subject librarians, dynamic database lists based upon users' access privileges, convenient integration with our content management system, and a simple backend management interface requiring little expertise to use. We then sought to improve the situation by organizational and license simplification before our 2014 website overhaul: We hired a dedicated licensed content librarian, and systems and technical services personnel worked tirelessly to streamline our seven privilege groups into two by negotiation and policy changes. We then modified our workflows to encourage this simplicity. This paper describes how technical services and systems can design and create an in-house application to ease the dynamic delivery of database information, authentication, and access. Then it describes subsequent improvements upon that system with organizational changes and better business practices.

When the first electronic resources came onto the market, licensing those resources began as an organic process at Mercer University Libraries, as is often the case. Our first licensing model mirrored our print acquisitions model exactly: selecting librarians would choose materials for their clientele, and the resources would be available for the clientele at their locations. The University Libraries subscribed to resources for two main campuses, which began with distinct programs: the Macon campus served a traditional undergraduate population, while the Atlanta campus served graduates and professionals. There are some satellite campuses, that we call the "Centers," which traditionally have served nontraditional students. The University also has a law school and a medical school, the libraries of which are administratively separate from the University Libraries.

Our University culture began with essentially three distinct and independent libraries tailoring electronic resources to serve their distinct programs, without much thought to the collective "University Libraries" institution. However, the greater University ecosystem was evolving beyond the location-bound programs: professional and

graduate programs spread to the Centers and Macon campuses, undergraduate disciplines appeared in Atlanta, and nontraditional programs appeared on all campuses. The marketplace was also evolving: new technologies like link resolvers and discovery services appeared on the marketplace, and we had neither the funding nor the personnel to manage three incarnations of each service. Additionally, there was considerable overlap between the pharmacy and nursing programs and the medical school's resources, which led to considerable cooperation across administrative boundaries. That demonstrated the feasibility of intercampus cooperation. By 2010, the time was ripe to figure out how to simplify our electronic resources.

Our library websites existed to serve our clientele, and funnel them to the content we provided. Although we shared an automation system, our websites were built with the electronic resources in mind, and as with those resources, we created three distinct web presences. The first step to simplifying our licenses became an implementation of a single website for three libraries with three distinct sets of electronic resources, and the major problem here was

creating the “database of databases” that would show everyone what they were able to access.

Feature List

We began with two different systems: one system based on a fragile Microsoft Access database, which was hosted on a SharePoint server and accessed via Active Server Pages, and a second based on a MySQL database, which was hosted on a Linux server and accessed via PHP scripts. We authenticated users with EZproxy using lists derived from library patron records in our Innovative Millennium system. These existing systems taught us what we did not like and how to do it better.

Building a system from the ground up gave us an opportunity to take a critical look at the shortcomings and build a list of features to avoid. Those included:

- Maintaining lists of users for our three user populations. We loaded patron records several times a year, and we had to update our lists of users for three different proxies each of those times.
- URL changes wreaked havoc on our websites. Each time a vendor changed URLs, we had to hunt for the database URLs throughout the site. This was tedious.
- EZproxy authentication was tough to explain to library personnel. It was nigh impossible to explain to the public. However, we had to manually create EZproxy URLs for the librarians to use.
- Neither database had a good management interface. Access gave us a tabular interface for data entry, and we managed MySQL with SQL commands. Neither of these was as good as a simple web form.

We identified some new features that would give us solutions to all of those issues, and give us the flexibility to solve our license organization issues until we had time to make our licenses universal across the university:

- Our campus had implemented ActiveDirectory, which was available for us to query. We should use this rather than reloading patron data into the proxy servers.
- We needed a web-accessible administrative interface
- Databases and patrons both needed to be classified by license group
- We needed to support persistent URLs
- Integrating this new system into our new content management system’s web editor would promote usage of the system

Authentication

Our remote authentication system was almost where it needed to be. We had already implemented EZproxy’s groups feature, so we had a start on classifying databases by license type within the proxy servers. Our old system relied upon users who already had accounts in the campus’s ActiveDirectory system. However, we were creating a shadow authentication system that relied upon the university ID number alone. The campus had not implemented a true single sign-on (SSO) system, but at least using ActiveDirectory would allow them to authenticate with the same username and password as they use all over the campus. We determined that the easiest way for us to do this would be with an external web application, and EZproxy’s ticketing system.

Since we had disparate licenses, we also needed to authorize our authenticated users. Restated, authorization tells us what privileges the user has. Authentication tells us that the user is who they say they are. Our campus Information Technology unit was rolling out a new student management system in 2010, so we had to create a new student data feed at that time anyway. We load that data into a PostgreSQL (<http://www.postgresql.org>) database. Because we get different data for employees than we get for students, we load the data separately into separate student and employee tables. We also

load this data into our library system. This is visualized in Figure 1 (see Appendix).

At its core, our remote authentication process is a simple process. It collects user authentication information (a username and a password) and queries ActiveDirectory to verify the credentials. After authentication, the system looks at our database to authorize the user. At this point, it constructs a URL with an EZproxy ticket, and sends the user on to the proxy server. The proxy takes over from there. This is visualized in Figure 2 (see Appendix).

We created this application using the CherryPy framework (<http://www.cherrypy.org>) and a few Python libraries to query the PostgreSQL database and ActiveDirectory. Additionally, we have processes that handle the daily ingest of user data from IT.

The Database of Databases

When looking at the features that we wanted to implement, we are really only looking at a single table of information. This needs to capture the following information:

- Database URL
- Database name
- *Brief* description
- Whether or not to use the proxy. This was to address the desire to add freely accessible databases to this system.
- License groups—initially this included the following groups: Atlanta, Centers, Macon, pharmacy, nursing and any combination of the above. We were later able to reduce that to two possibilities.

We investigated a number of technologies to implement our new database of databases, and we settled upon the Django framework (<https://www.djangoproject.com>). Django gives us a robust administrative interface for next to no code, once we have implemented our data model. The implementation of the model only required 22 lines of code. We made a number of design

decisions that would prevent problems that arose with our previous systems. Chief among those were:

1. Limit description to 512 characters. We could write novels about each resource, but users did not look for that level of detail in this database of databases list. That should be done in a library guide somewhere else.
2. Make the database URL a unique identifier: it is hard to manage URL change when we have dozens of different names for the same resource.

Refer to Figure 3. (see Appendix) for a screen shot of the administrative interface for the data model.

We deployed a series of views that show users' different versions of this data to produce different functionality on the website:

- An A–Z list of the databases that we show on our home page for quick browsing—see Figure 4 (Appendix).
- The actual browse list, with the databases listed alphabetically—see Figure 5 (Appendix). Note the licensing restriction in Figure 6 (Appendix).
- A view that handles persistent URLs. This view shows no content, but rather it looks at authentication, authorization, and the database classification to either redirect directly to the resource, or through a proxy server.

A key function of this system, especially when our licenses were so complicated, was to filter the content so that each authenticated user would only see resources to which he or she was entitled. We also had to build functionality in for manually curated subject lists, which is included in the home page widget. This is not functionality within the application, but it is derived from the same data.

Our persistent links are a key part of this system, and we guaranteed to our users that these will never change but will intelligently use the proxy,

and because of that they will work everywhere without making our users understand when to use the EZproxy prefix. For an example, visit: <http://libraries.mercer.edu/api/quicklink/2>

Website Integration

In 2010 we implemented the Plone (<https://plone.org>) content management system (CMS), which used the user-friendly TinyMCE web editor so our users no longer needed to know any special markup. However, we anticipated some issues: librarians frequently do not know which URL is the appropriate URL, and it is hard to educate content creators about the Proxy.

Integrating the database of databases into the editor was an important usability feature. A simple “DB” button in the editor invokes the system—see Figure 7 (Appendix). The application window has browse and search tools, and once records are retrieved, the content creator can insert just the link into the database, the link and the description, or everything including the access statement—see Figure 8 (Appendix). The resulting text appears in the editor window.

Relicensing and Workflow Simplification

As we previously pointed out, we knew that our licensing situation was unsustainable. However, setting things right would require time, and thus some interim steps were necessary. The database of databases system bought us time to relicense our resources to address the fact that our programs have spread out across the university’s physical locations, and create nearly universal licenses so nearly every student could access nearly every resource.

The primary issue was that too many people had their hands in the licensing task. This created confusion for our vendors, as no one knew which person to talk to, and each person had a slightly different perspective. We created a single position, the Licensed Content Librarian (LCL), to handle all pricing and licensing discussions and be a single voice to all of our vendors. The second task was to charge this new librarian to renegotiate all of our licenses to cover as many students as possible. This was time intensive, and

it cost additional money. However, with careful analysis, we were able to shed some redundant resources, and nearly cut the initial estimate in half.

The net result of this position was greatly simplified licensing after nearly two years of work. We ended up with only two license variants, and only three of our 212 databases have any program restrictions placed on them.

A second prong of this reached into the future: we needed to create a stable, forward-thinking workflow for electronic resource licensing. The resulting workflow is beyond the scope of this paper, but it included the following phases: discovery, evaluation, budget, licensing, access, and launching. The LCL was central to every phase of this process, and interacts with all players from selector to end user. The policies also have teeth to ensure that we do not regress to our previous state of affairs: there is the possibility of disciplinary action if personnel attempt to license resources outside of the workflow.

The 2014 Website Update

With our licensing issue solved, we were motivated to evolve the website to a new point. The old site was location centric, and it had four distinct personalities: one for each campus, plus a “University Libraries” personality for off-campus users. The intent when we implemented it was to merge three separate websites into a single website, but we were not able to make many content changes. This compromise caused our home page to be slow—it was buggy and our users frequently got lost in the complexity. The solution was to make the website function-based, rather than location-based. This was mainly a reorganization of content, with a few application twists.

This redesign prompted some updates to our database of databases application: we could now replace the “availability” statement in our database A–Z listing, because virtually everything was available to everyone. It was redundant to say that everything was available, so we removed it, and only disclosed the three resources with access limitations. With the simplified licensing We also

had begun the implementation of the EBSCO Discovery Service (EDS), so we needed to classify our resources based upon whether they were indexed by EDS, and we determined that we needed to indicate which organization or individual funded each resource. This added a minor amount of complexity to the data model.

Conclusion

The implementation of this database of databases application taught us quite a bit in a number of areas. The most important lesson learned was the primacy of requirements gathering: when one knows what functionality needs to be

implemented, and one knows the problem very well, one can create a robust application. It was very telling when preparing this paper that most of the code that we wrote in 2010 was still there, untouched because it continues to work well half a decade later. We also learned that a complex problem creates complex code, which in turn creates user confusion and bugs. However, this became one of the talking points for our effort to advocate for our licensing simplification project. When that simplification project succeeded, it led to a great simplification in workflows and allowed us to remove hundreds of lines of code from the database of databases web application.

Appendix

Figure 1. A visualization of the flow of user data from the student system, registration system, and human resources into data files into the library systems. Once the library takes hold of the data, we load it into the database, and then create a patron file for the library system. Our remote authentication process looks directly at the database.

Figure 2. A visualization of the authentication and authorization process.

The screenshot shows the 'Change database' form in the Database of Databases interface. The form includes fields for 'Database URL' (Currently: <http://site.ebrary.com/lib/mercer/home.action>), 'Database Name' (ebrary Academic Complete eBook Collect), and 'Database Description' (A multidisciplinary collection of over 125,000 scholarly e-books). There are checkboxes for 'Use Proxy?' (checked) and 'Restricted to Nursing & Pharmacy' (unchecked). Red arrows point to the 'Use Proxy?' checkbox with the text 'Use Proxy?' and 'License Restriction'. The form has buttons for 'Delete', 'Save and add another', 'Save and continue editing', and 'Save'.

Figure 3. Database of databases data entry, where we capture the URL, name, a short description, whether to use the proxy, and finally which license group. Note: this is an interim stage, where we have already simplified our license choices down to one.

Figure 4. An A–Z browse list for our home page.

Figure 5. The A–Z listing of each database.

Figure 6. An access-limited resource. Note: in our 2010 implementation, this access statement was on each resource. Now we only show it when a resource is limited.

Figure 7. The DB button in the editor toolbar invokes the system.

Insert Database Link

SearchTerm:

Browse By: [A](#) [B](#) [C](#) [D](#) [E](#) [F](#) [G](#) [H](#) [I](#) [J](#) [K](#) [L](#) [M](#) [N](#) [O](#) [P](#) [R](#) [S](#) [T](#) [U](#) [V](#) [W](#)

Search

Cancel

1 hits for 'lexicomp'

Lexicomp Online — Drug monographs for prescription and OTC drugs, includes toxicology, brand names, and medication safety issues.

Available to: Nursing, Pharmacy

[Insert Link only](#) | [Insert Link and description](#) | [Insert everything](#)

Figure 8. The database integration window.