Leveraging Drupal Capabilities to Create a User-Focused Library Website

Anne M. Platoff, Ian Lessing, Mai Irie
University of California, Santa Barbara Library -- http://www.library.ucsb.edu
Presented to Internet Librarian 2012 (Monterey, California), 22 October 2012

Abstract

Redesigning the website for a major academic library is never an easy task. It is even more complicated when combined with the migration of a static HTML site into a content management system. The UCSB Library launched its Drupal-powered website in August 2011. Throughout our redesign project we concentrated on creating a user-focused site which not only provides information about the Library, but also serves as a gateway to research for users with a wide variety of information needs and research literacy. We selected Drupal as our content management system. In our presentation, we will explain the variety of Drupal modules and capabilities that we have used to create a user-focused site. The discussion will include the technology stack behind the site, the build process, the management of content, and how we leveraged Drupal capabilities to create specific features for our site. Topics discussed will include our database of electronic resources, enhanced site search, assortment of targeted news feeds, “smart” help section, feature boxes, and performance issues. The UCSB Library Drupal site is located at http://www.library.ucsb.edu.

Overview of the Site and the Project

The launch of our new library website for the Fall 2011 quarter marked a major milestone in a multi-year effort. Our project achieved multiple library goals including the revision of most website content, migration into the Drupal content management system (CMS), a complete redesign of the site, and creation of new website tools for our users. We selected Drupal as our content management system because it is open source, allows for flexibility and custom coding, and because there is a growing community of Drupal users on our campus and worldwide. In addition, there are no licensing costs associated with using Drupal, so the expenses incurred are limited to developer hours.

Our library systems are hosted on a VM Ware-based virtual system infrastructure. This allows us to have multiple virtual machines running on one or more central servers utilizing a large storage array. This is an enterprise-level system that may be out of reach for many libraries, but a very similar system could be built using Amazon’s cloud services which are very reasonably priced. We are running Ubuntu Linux version 10.04 for all our servers. The production server has 2 CPUs allocated, 4 GB of memory, and 7.5 GB of disk space. This server generally utilizes about 6% of the CPU, 1 GB of unused memory, and 2-3 GB of free disk space. It runs standard versions of all the software involved —- Apache, MySQL & PHP. We have not done extensive performance tuning of these software components. We also have a separate SOLR server with just 1 CPU allocated and 2 GB of memory. We developed the site using Drupal 6, as Drupal 7 was still in beta when we started the project. For now, we have no plans to upgrade for at least a year. For a list of contributed modules and their functions, please see our additional handout titled “UCSB Library Drupal Site: Modules, Libraries, and Themes”. Although we rely mostly on contributed modules for additional functionality, we do use some custom modules for UCSB Library-
specific features. These features include the customized search interfaces as described in the Database of Electronic Resources and Enhanced Site Search sections.

We use a four-tiered system for development, feature review, and content editing. Our Dev site is a local instance used for development work on new features and bug fixes. Content editors do not have access to this site. Our Preview site is used primarily for training content editors. It is also the place where new features are initially rolled out for feedback and review before being implemented on our staging site. Content on the Preview site is not part of the promotion path for the live site. This allows for a safe place for content editors to experiment and test, knowing that whatever they do there will not be seen by the public. The Stage site is where content editors do all their content creation and editing. Lastly, there is the Production site. This site is the public-facing site and only our Site Managers have editing privileges on this instance. Content is typically not altered on the Production site, and if it is the same changes must be made on both Production and Stage. Additional performance caching (Boost) is also used on Production.

Using the Workflow module, we defined two states for content: Draft and Public. If content is in the Draft state, it will not be made visible to the public on our Production server when promoted to the live site. Only content in the Public state will be moved and made visible. We developed our own system for promotion of content from Stage to Production. This is a multi-step process that requires a trained privileged technician using Linux command-line tools. The process takes about 10 minutes and the site stays active and functional throughout the process. This process is only necessary because we have separated our Stage site and our Production site. On many Drupal sites content editing is performed directly on the Production Drupal site and therefore no content promotion is required.

Advantages and Disadvantages of Drupal

There are countless advantages to using and developing a site with Drupal, with the most important being the flexibility in features and the power that content editors enjoy with such a robust system. The huge repository of contributed and maintained modules, as well as the many distributions (full copies of Drupal that include Drupal Core, along with additional software such as themes, modules, libraries, and installation profiles), provide an easy way to get up and running quickly. Partly because of its modularity and flexibility, there are several performance issues that affect Drupal sites. Thus, it is important to install only the modules that one actually needs. It is also critical to install caching modules such as Memcache, Boost, Varnish, Advanced CSS/JS Aggregation, etc. to improve performance.

Although the learning curve for tackling module or theme development is quite steep, a developer or themer can learn both topics from the large and helpful Drupal community and by looking at the code for Drupal Core, or a contributed module or theme. The Drupal community also makes a serious effort to provide clear documentation, tutorials, and examples.

Site Design and Theming

Early in the project it became clear that we did not have the Drupal expertise in our staff to do all the work required in a reasonable time frame. So we decided to hire a Drupal consulting firm to assist us. Our Theming skills were thin so we found a firm to work on the visual design, information architecture, and Drupal-specific aspects of the site. After a lengthy R.F.P. process we selected an experienced and
reputable vendor and embarked on a cooperative effort with them. Like any vendor relationship we had some ups and downs. We found their experience and knowledge of website design and information architecture very valuable. Their early mockups of the 3-5 most important page types of the site helped us make the hard decisions about site structure, menus, hierarchy of navigation, and other issues that had to be settled before serious site creation work could commence. In later stages of the project our Drupal consultants got very busy with other clients and we found it difficult to get timely responses and work output from them. There were also budget over-runs and communication difficulties because our contractor was located in another state. On the whole, though, we believe the money used for outside consultants was well spent as it gave us a good start on the project.

While our redesign was in process, we successfully recruited and hired a Drupal developer who also had experience with theming. We have found that having our own staff member with these capabilities has allowed us more flexibility in tweaking our theme, and also has been vital to the development of site tools and enhancements. Contractors are great for the initial work on a site, but when you are developing your own custom features or are dealing with unexpected issues, there is no substitute for having someone with Drupal expertise on your staff.

Our site theme is a subtheme of NineSixty (960 Grid System). This theme is based on the 960 Grid system by Nathan Smith, an effort to streamline web development workflow by providing commonly used dimensions, based on a width of 960 pixels. We use Rubik for our administration theme. Content editing uses our site theme rather than the administration theme. For more information, see our handout entitled “UCSB Library Drupal Site: Modules, Libraries, and Themes”.

**Database of Electronic Resources**

One of our primary goals in our site redesign was to develop a database-driven tool which would simplify management of our research databases and enable us to provide users with enhanced discovery methods for these resources. This tool, which we often refer to as our “database of databases”, allows us to manage links to subscription and free resources that we recommend to our users. At this time we do not have a centralized system for managing all e-resources, nor do we have a federated search tool that includes all of our online materials. Our electronic journals access is managed using the California Digital Library’s shared SFX system and our electronic books are included in our online catalog. Prior to our redesign, the database links were maintained manually on multiple subject-based HTML lists. When a database title or URL changed, we had to make the changes on every list where that database was listed.

To create our database of databases we established a special content type in Drupal for research databases, with fields for all the information we needed about an individual database. We also created several Drupal taxonomies (controlled vocabularies) which allow us to assign multiple terms from each taxonomy list to a database. Our taxonomies cover: Subject (e.g., Film Studies or Geology), Broad Category (e.g., Engineering or Humanities), Materials Indexed (e.g., Book Chapters or Journal Articles), and Database Type (e.g., Audio Collection or Full Text Collection). Additional information collected for each database includes title, alternate titles, search URL, mobile search URL, database help URL, database availability (subscription or open access), dates of coverage (specific date ranges or thematic dates like “American Civil War era”), a database description, links to related databases, identification of print equivalents, and interface/materials languages. One database license requires us to IP-
authenticate the instructions for setting up an account, a request that we have been able to accommodate within our database design.

We produced a few search interfaces to allow users to browse and locate a resource. These are all created using a single View with multiple displays. A custom module is used to customize some of the theming of the View output such as the clickable icons, and to set up the block content for our tabbed search box. The Contexts module is used to define the blocks that should appear on certain View Page displays.

The first interface is the search tab that has the following options: Find a Resource by Subject, Find a Resource by Category, and Find a Resource by Title. The Find a Resource by Subject and Find a Resource by Category options are generated by defining two Block displays in the view. Each has an exposed filter displayed as a dropdown list of terms from the appropriate taxonomy's vocabulary (Subject or Broad Category).

To produce the Find a Resource by Title section we used an Attachment display. This display shows a clickable alphabetical list (Alphalist) for the database titles. The Alphalist is actually created by using an Argument for “Node: Title” with “Glossary mode” selected. When a user clicks on a letter (or 0-9), a new page loads which shows a modified search tab (with only the Find a Resource by Subject and Find a Resource by Category options), the clickable Alphalist, and a list of database titles that start with the selected letter or number. This page of results is handled by another Page display in the same View. It simply expands on the display setup by the Alphalist attachment. We linked the database title to the database URL and offer 2-4 clickable icons after the title. All databases in the list include a magnifying glass icon linked to the database URL and an icon with the letter “i” in a circle which links to the information page (a page displaying all information we have in the database of databases for that title). In addition, research databases with mobile versions have a mobile phone icon linked to the mobile version, and subscription databases have a lock icon indicating that access is restricted to UCSB users.

One of the reference librarians’ favorite aspect of this tool is the ability to create a “Most Recommended” list for each subject area. To do this, we use Drupal nodequeues (specifically, the “taxonomy queue”) to assign “Most Recommended” status for a particular subject and to customize the order in which these databases are presented to the user. When a user selects a Subject search term such as English, the “Most Recommended” databases for the subject are shown at the top of the results in the order dictated by the subject librarian. Following this are all databases to which we have assigned the subject term in alphabetical order. This page is controlled by a Page display in the View. The Most Recommended section is handled by an Attachment display.

Enhanced Site Search

Prior to the launch of our Drupal site we were relying on Google indexing for our site search. Drupal has given us more flexibility in how we handle site searching and present search results to our users. Our current site search feature not only returns all the instances of the search term in the results, but also shows how many results are in each section of the website. Users can “filter by section” and narrow their results down more easily. In addition, users who misinterpret the purpose of our site search box and try to use it as a federated search tool will receive a helpful instructional “null results” message that says “Your search term x was not found on the library website” and then provides the user with
guidance and links to the library catalog, the search & research page, the website help section, and our Ask a Librarian service.

To operate our site search feature we are using Apache Solr, an open-source enterprise-level search platform from the Apache Lucene project. It is written in Java and runs as a standalone full-text search server. The Apache Solr Search Integration module provides an interface for Drupal to send content to be indexed, and to submit search queries to this separate Apache Solr instance.

Our faceted search is accomplished by enabling the Apache Solr Search: Filter by Taxonomy section filter. Using the Context module, we can display the Filter by Sections block on the left side of the search results. The Current Search block showing the number of results, the search terms, and any selected filters are also provided by the Apache Solr Search Integration module.

We also created search tabs for: All, Special Collections, and News content. When a user clicks on the News tab, the search results are filtered only to News content. This tabbed interface was accomplished using a custom module. The module added these search URLs to the menu and expanded the search function to include any search terms and selected filters (as appropriate) when switching tabs. The module also created the custom null results message.

**News Feeds**

Our homepage design includes a news feed box with blog stories we post for our users. Before moving to Drupal, we were importing in headlines from a hosted WordPress blog and linking users over to the blog to read stories. This presented many problems for us including inconsistencies in site design and navigation, as well as the unfortunate insertion of ads at the bottom of each news story. With Drupal, we have created a content type for news stories. Blog entries are contained in the database and are now included in our site search results. In addition, since they are managed within Drupal we now have our site header, footer, and navigation on all our news stories.

Perhaps one of the best aspects of having the news blog in our main site is that we are able to create separate blogs for different sections of the website. We use nodequeues in Drupal to assign stories to different news feeds, and because stories can be assigned to multiple queues we are able to share stories between multiple feeds. Using separate nodequeues, Views, and Context gives us the flexibility to have certain news blocks display for different sections of our site. These news blocks are created using a single View with multiple Block displays. RSS feeds allow our users to subscribe to content from the different blogs. The RSS feeds are also handled by this same View using the RSS display. The clickable News & Events block header is accomplished via a preprocessing function in the theme layer.

**“Smart” Help Section**

Anyone who has designed a library website, or tried to instruct users in how to navigate such a site, knows how complex these library websites can be. They are not only a source of information about the library and library services, but they also function as a gateway to numerous library resources such as library catalogs, subscription databases, e-journals, and other full-text content. Users not only have to contend with a complex site layout, but also often lack expertise in how libraries organize information and how to effectively conduct library research. For this reason, one of the most important sections of our site is our Help section. During our redesign, we rewrote and expanded our help pages. What we
really wanted to accomplish in our migration to a content management system and our site redesign was the creation of a user-friendly tool to get users the help they need without overwhelming them with search results. Think about a user who just wants to know how to find books in the library. If they were to go to our website search box and enter the word “books” they would find 615 pages with this term in their search results. This would be a bit much, especially for a user who is already frustrated. What we have done for our help section is to create a special “help” taxonomy of terms which are assigned selectively and only to pages we want to classify as help pages (they do not have to be in the help section to be assigned help terms). Users of the help section have two options – they can either browse by these taxonomy terms to find just the content to which we have assigned those terms, or they can use a search box. When they search, Drupal only returns links to pages that have been designated as “help” pages through the assignment of a help taxonomy term. This search returns a significantly smaller list of results and makes it much easier to navigate to the type of help they need.

Both the search dropdown and the search box are controlled by a View. The dropdown is a Block display with an exposed term filter that queries content tagged with a help taxonomy term. The search box is part of the Page display that has both the Search filter and the list of search results (if a search has been performed). On both the help page and the help search results page, we provide them with easy access to our Ask a Librarian service in case they need additional help.

Feature Boxes

Another content type that we have created for our Drupal site is used for the entries in our feature boxes, two of which are shown on our library homepage. “Library Highlights” are in a larger content area which displays as a slide show and include images, links, and photos related to collections and hot topics. A smaller content area houses our “Know Your Library” section which includes information about library services and resources. The third feature type is displayed in a “Featured Database” box on our database pages. All three feature box regions are populated with the same content type. We use nodequeues to assign items to different boxes. This type of arrangement allows us to share content between “Library Highlights” and “Know Your Library”. We can also make the pool for the slideshow smaller by simply removing the item from the nodequeue, while leaving it on the site. This allows more flexibility with the content in these areas. We use a View to create different block and page displays for these feature boxes. The slideshow effect is a view style provided by the Views Slideshow module.

Performance Issues

One drawback that should be known about Drupal is that it is not super fast and has performance issues. The performance issues are not insurmountable and certainly there are many very high-traffic sites built using Drupal. But it is important to understand that a basic Drupal system will begin to respond poorly when it has to serve a lot of concurrent connections. Before releasing our site we load-tested our Drupal system with a Linux tool called Siege. This test showed us how poorly our system would function with 30, 90, 150, or more connections. We recommend this exercise to anyone building a Drupal site for public use. It is also important to understand that there are various “Distributions” of Drupal. We use the PressFlow distribution, which is tuned for higher performance than the stock distribution.

Luckily there are some tools and techniques which can be used to improve Drupal’s performance. Following are the ones we use, but others are also available. Memcache is a database-layer caching
mechanism holds query results in a cache so that subsequent identical queries can be answered by
drawing from the cache rather than calling from the MySQL database that powers Drupal. For example,
homepage elements do not have to be assembled by the database every time the homepage is
requested by a browser. Of course, this only comes into play if a page isn’t already in the boost cache,
which is generated by Boost module. Boost module is a page-layer cache that holds fully-formed HTML
content in a file on disk. The advantage this yields is that Drupal doesn’t have to fully start up, query the
database, and then assemble all the content from various storage areas. It also reduces server load,
freeing the server up to answer queries for pages that are not cached or to perform other tasks.

We did have to sacrifice one feature in favor of performance gains. The library subscribes to proprietary
databases which are available only to University-affiliated researchers within our campus IP ranges. In
order for our faculty, staff and students to access these databases from an off-campus location, they
must use our proxy server. For a large segment of our patrons the details of when and how to use the
proxy server can be somewhat confusing. So we originally planned to develop a feature whereby Drupal
would check the user’s IP address, determine if the user was off-campus, and then would automatically
add a prefix to database URLs that would route the user through the proxy server. On-campus users
would be given a non-prefixed URL, and would therefore bypass the proxy login. The problem we
encountered in offering this feature occurred because of our Boost page-caching. If an off-campus user
was the first user to visit a page, then subsequent users (both on-campus and off-campus) who visited
that page would be directed to the proxy even though it was unnecessary for many of them. Likewise, if
the first user was an on-campus user, all users (both on-campus and off-campus) received the non-
proxied link, basically breaking the feature for subsequent off-campus users of that page. Turning off
the Boost cache would have slowed page-load time to unacceptable lengths so we decided to turn off
the automatic proxy-routing feature until a work-around can be devised.

In addition to the database and page layer caching mechanisms, we also aggregate, minimize, compress
and cache CSS and JavaScript files. The JavaScript and CSS files combined from Drupal Core, themes and
contributed modules results in a large number of files that must be downloaded by the user on a given
page load. Using the Advanced CSS/JS Aggregation module, Drupal will combine all the CSS files into
one Gzip’ed compressed file. The same applies to the JavaScript files. This can make the page load
to faster and additionally reduce server load a bit.

**Conclusion**

Overall, we have been quite happy with the results of our redesign project and conversion to Drupal.
While the project took a long time and was not always easy, we feel that the benefits to the library and
to our users have been well worth the effort. Our website has moved from a collection of static HTML
pages with disparate designs, to a database-driven system where we are able to apply a common look
and feel, as well as standard menus, to all pages regardless of where they reside in our site. Content
editors can more easily update their pages because they can use a WYSIWYG editor and do not need to
know HTML. Site managers can spend their time creating enhancements to the site, rather than
checking the work of the content editors. Our website content is also much more manageable now that
it is in a database. In addition, we know that future redesigns will be much easier because the content
management system will make it easier to reorganize our content and will enable us to easily apply new
themes across the site. In addition, Drupal has given us the power to continuously upgrade and create
new features, thus improving services for our users without major redesigns.
One of the biggest advantages of using Drupal is the flexibility it gives in overall web management and development. For example, if you use Drupal, not all your content has to live in Drupal and not all your services and functions have to be Drupal modules. Drupal can easily pull in content from other sources such as Twitter, Facebook, an XML feed from any source, etc. While Drupal Core provides a Search module, we chose to use Apache Solr Search. We use the Apache Solr Search Integration module to provide the bridge between the two applications. In addition, not all your content has to be accessed on your Drupal site. You can use the Services module to create web services via multiple interfaces like REST, XMLRPC, JSON, JSON-RPC, SOAP, and AMP for consumption by a remote application. While how to achieve this is beyond the scope of this paper, you can learn more about it at http://drupal.org/documentation/modules/services.

Finally, Drupal is an excellent example of the power of the open-source software development model. As the Drupal website notes, it is maintained and developed by a community of 630,000+ users and developers. This means that it is much more dynamic than proprietary content management systems. Its modularity allows for continual development of new features and presents more options for site developers. The Drupal community is continuously expanding and support resources, events, conferences, and user groups are exponentially growing. This allows Drupal developers many opportunities for professional development and information sharing. Overall we feel that this makes Drupal an excellent choice for libraries seeking a dynamic and flexible platform for managing their websites and offering enhanced web services to their users.

References and Resources

Drupal Resources and References

- Drupal Website: http://drupal.org/
- Community Drupal Documentation, http://drupal.org/documentation
- Pressflow Drupal Distribution, http://pressflow.org/
- Siege, http://www.joedog.org/siege-home/
- IRC Channels: http://drupal.org/irc
- Drupal Marketplace (Services, Hosting, Training): http://drupal.org/drupal-services
- In Person Community Resources (Groups/Meetups):
  - Drupal Groups: http://groups.drupal.org/groups
  - Local Meetups and Playdates: http://www.meetup.com
- Mailing Lists: http://drupal.org/mailing-lists
  - Be sure to sign up for the security announcements -- http://drupal.org/security and the Drupal Association -- https://association.drupal.org/ for up to date information about Drupal news, grants, free training, Drupal Camps, and events as well as other exciting topics.
- LinkedIn Drupal Group: http://www.linkedin.com/groups/home=&gid=35920&trk=anet_ug_hm
- Some Online Tutorials for Drupal (there are too many to list)
  - Recorded DrupalCon Sessions: http://drupal.org/drupalcon (also do an online search for DrupalCamp sessions -- sometimes they are recorded as well)
o NodeOne: http://nodeone.se/sv/learning-library
o Lullabot: http://www.lullabot.com/
  o Learn By the Drop: http://learnbythedrop.com/

UCSB Library Website and Discussed Features

- Main Public Website: http://www.library.ucsb.edu
- Database of Databases: http://www.library.ucsb.edu/research/resources/databases
- News & Events: content box on homepage and full news feed at http://www.library.ucsb.edu/about/news
- Departmental News Feed for Special Collections: on Special Collections homepage at http://www.library.ucsb.edu/special-collections and full news feed at http://www.library.ucsb.edu/special-collections/news
- Faceted Site Search with Tabs: search box at upper right on home page and also at http://www.library.ucsb.edu/search/
- Smart Help: http://www.library.ucsb.edu/help
- Library Highlights: feature box on library homepage, full list of entries at http://www.library.ucsb.edu/highlights
- Know Your Library: feature box on library homepage, full list of entries at http://www.library.ucsb.edu/know-your-library
  - Featured Databases: feature box on database pages, full list of entries at http://www.library.ucsb.edu/featured-databases

About the Presenters

Anne M. Platoff, Web Services Librarian
platoff@library.ucsb.edu

Annie is the web services librarian at UC Santa Barbara Library. She provides user support to people using the UCSB Library's online services and manages the content of the library website. Annie has been managing library websites for 15 years. She is interested in how libraries can design their websites to serve a variety of user types with various information needs.

Ian Lessing, Software Development Manager / Web Programmer
ilessing@library.ucsb.edu

Ian is a web programmer and manager of the programming & ILS unit in the UCSB Library's IT dept. He works with a variety of different web technologies and has a background in e-commerce and software development projects.

Mai Irie, Computer & Network Technologist
mirie@library.ucsb.edu

Mai is a web developer at UC Santa Barbara Library. She has a background in both graphic design and web development, and is a graduate of UPenn's Master of Computer and Information Technology program. She works mainly on enhancing and maintaining the Drupal library website.