Chapter 4

Building the Branch

Abstract

Thus far, this issue has discussed how to plan your digital branch. This chapter of "Building the Digital Branch: Guidelines for Transforming Your Library Website" examines the process of building the site, and provides guidelines and resources. The author discusses how to select a content management system, how to determine what type of server setup might work best and best practices for site design.

So far, we have gathered information and planned what to build from that information. This chapter discusses more some of the "hows"—how we built Topeka's digital branch and what the process can teach us. We'll look closely at what to consider during the building phase of your digital branch project.

First Decisions

Before you start building, you have to make a few decisions about the tools you plan to use. By tools, I mean the software and coding languages that are going to hold your digital branch together. Some of these include what content management system you will use and what type of server you will use to host your digital branch. Also assess whether or not your IT department has the skills in-house to handle hosting and supporting the software, servers, and tools you decide to use.

Ultimately, you want to make your decisions based on the goals for your library's digital branch, not on the skills of your staff. Remember, you can always train staff to support a new server. These days, it's also easy and can be quite cost-effective to outsource that part of your website. You can certainly change this type of infrastructure after the fact, but it will be a lot of work. Get it right up front, and you will potentially save yourself and your library years of time and work.

Content Management System

Let's talk a little about content management systems (CMS). Your library's CMS will be the heart of your new digital branch. Your Web team will take whatever CMS you pick and will design the digital branch on top of it. They will base all future decisions on and around the CMS. Anything they add later on will need to be based on the functionality, features, and expandability of this CMS. So it really behooves you to get this decision right very early on.

The CMS is the control center of your digital branch. It allows you to administer content within the branch: how it is organized, displayed, and accessed; who can access it; and how it can be accessed. According to one definition, a CMS is "a tool that enables a variety of (centralized) technical and (de-centralized) non technical staff to create, edit, manage and finally publish (in a number of formats) a variety of content (such as text, graphics, video, documents etc), whilst being constrained by a centralized set of rules, process and workflows that ensure coherent, validated electronic content."[1]

A CMS will make your library's Web life much easier. Most of your library's staff will be adding content to the digital branch, but they probably won't have advanced HTML or CSS skills, and they probably don't have a web-

site editing software package like Dreamweaver installed on their work PCs. With a CMS installed, all those staff members need to add or edit content for their digital branch is a Web browser. In fact, adding content will be similar to creating an e-mail using a Web-based e-mail solution like Yahoo or Gmail. Most CMSs use automated, built-in templates. Once these templates are set, your staff has to worry only about content–the "look and feel" is already created.

Since a CMS is Web-based, it will be much easier for your techie staff to deploy. All they have to do is train library staff to use the new tool (which will be easy–it will be the "Go here, click this button, start typing, press Publish" type of training). Once you have a CMS in place, anyone who can type in a box and press a button will be able to add content to your digital branch.

Adding new content is easy and can be done through a Web-based form. This page will look very similar to e-mail or Microsoft Word. The user basically types in a box, then presses the Publish button. The CMS will have many of the normal features that you find in other boxes you type in–bold, italic, underlined text, a way to add a Web link, etc. Once content is published, if you need to go back and edit it some more, a CMS makes this easy as well.

A very important point for libraries–there's really no reason to spend lots of your library's money and purchase a CMS. Many of the popular CMSs used today, even by large corporations, are free, open source, well-supported and well-documented software, so no up-front costs are attached.

What Do You Want Your CMS to Do?

The functionality of different CMSs varies. If you are building a digital branch, these functions are essential:

- *RSS*–You will want your CMS to have RSS feeds built in.
- *Articles or blog posts*–Most likely, you will want a CMS that can post articles just like a blog does. In fact, multiple blogs with multiple RSS feeds can give you even more options for content. These can also act as stand-alone webpages.
- *Comments*–You'll probably want a CMS that can handle comments. Create comment boxes on the front end that match up with content on your site and have a way to moderate, edit, release, and delete comments on the back end. Even better is a "naughty word" blocker so you don't have to constantly search for those seven bad words your board or administrators really don't want to see on your digital branch.
- *Ratings*–It might be nice to rate each article, event, etc. A one- to five-star rating system does this nicely and is found on many a CMS.
- *Categories*–It is useful to be able to assign one or more categories to an article.
- *An easy way to edit pages*–You'll want this on the back end. You don't want to force nontechnical staff to learn advanced XHTML and CSS (though it's helpful if they DO know these things). Instead, it's much better to have them log in, click the New Post button, and start typing.
- *Multiple logins at different levels*–Nontechnical staff should be able to log in easily to their part of the website to post content, but not have access to back-end settings. The feature of different user accounts (e.g., user, editor, and administrator) is handy.

CMS Options

There are many CMSs to choose from. Here are a couple of the more popular ones, plus the one my library is using at the moment.

Drupal

Drupal is a free, open source CMS written in PHP (see figure 9). Many types of websites use Drupal, from small personal blogs to large corporate websites. Drupal's features include the ability to register and maintain individual user accounts, administration menus, RSS feeds, customizable layout, flexible account privileges, logging, a blogging system, and an Internet forum.

Drupal
www.drupal.org

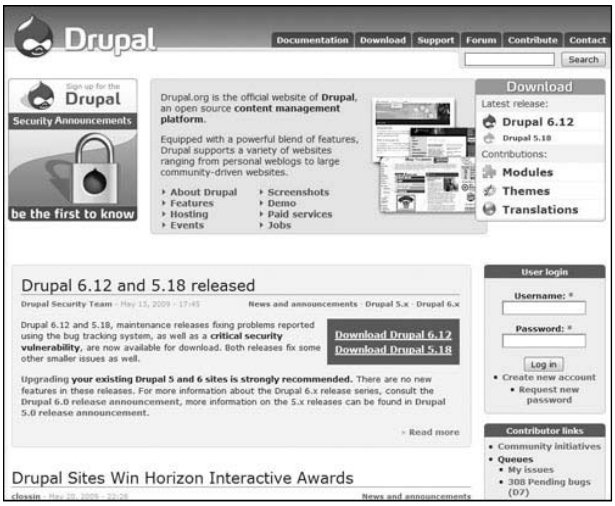

Figure 9
Drupal homepage.

Drupal was designed to allow new features and custom behavior to be added by third parties. Although Drupal offers a sophisticated programming interface for developers, no programming skills are required for basic website installation and administration.[2]

Joomla!

Joomla! is another free, open source CMS (see figure 10). Joomla's features include page caching, RSS feeds, printable versions of pages, news flashes, blogs, polls, website searching, and language internationalization. MTV, Citibank, and Harvard University are among some larger sites using Joomla.[3]

Joomla!
www.joomla.org

Figure 10
Joomla! homepage.

ExpressionEngine

ExpressionEngine is a commercial CMS designed by EllisLab (see figure 11). ExpressionEngine is similar to other CMSs in terms of features–RSS, blogs, commenting, different types of tracking and logs, and the ability to build membership-only communities are a few of its features. It's not an open source product, so it does cost–but not much. Currently, a personal license is $99.95, and a commercial license is $249.95.[4]

ExpressionEngine
www.expressionengine.com

Figure 11
ExpressionEngine homepage.

Topeka and Shawnee County Public Library uses ExpressionEngine, as does the Sundance Film Festival 2009 website and Roy Rogers Restaurants, as did President Obama's Change.gov site, which was his transitional site as President-elect.[5]

CMS Resources

*Wikipedia's "List of Content Management Systems"**
http://en.wikipedia.org/wiki/List_of_content_management_systems

OpenSourceCMS

(Web-based service that allows visitors to test a variety of CMS products online)
www.opensourcecms.com

**(includes both open source and proprietary CMSs)*

Windows or Linux?

As you look through a large list of CMSs, you'll notice something–they're written for different types of servers. Some are written primarily to be installed and run on a Microsoft Windows–based server; others are written to be installed and run on a Linux-based server; and some work on both platforms. You now have a lot of options and an important choice to make.

One crucial consideration is your IT staff, or whatever IT resources you may have on your staff. Can they handle the CMS you choose? Will they be able to learn it? As you are picking a CMS for your website, think about these things:

- What does your data center look like? What are your servers? Are they primarily Windows-based servers? Will a different type of server fit into your maintenance and backup and storage plan?
- Are your IT system administrators able and willing to learn new things? Many libraries are set up with Microsoft products primarily on the back end, and the system administrators have little or no training in a Linux-based system. If you choose Linux, they will have to learn how to manage a different type of server.
- What different coding languages does your CMS use? Does your IT/Web staff know that language? If the CMS runs in PHP and stores things in a MySQL database, can your staff deal with that?

For the most part, the modern Web—even large sites like search engines and stores—runs on a mix of open source, JavaScript, and PHP. This generally means using a flavor of Linux-based servers is best. I suggest going with that, and, if necessary, training your technical staff to use and maintain a new type of server. Of course, like most aspects of building a digital branch, much depends on your circumstances and individual preferences.

Next Up: Design

You may be tempted to start working next on programming your site, but I've found that doing things this way can put you at a disadvantage. Your next step should be design. Here's what the group 37signals has to say about building Web applications, and these comments apply to creating your digital branch:

> Too many ideas for websites start with a program-first mentality. That's a bad idea. Programming is the heaviest component of building an app, meaning it's the most expensive and hardest to change. Instead, start by designing first.
>
> Design is relatively light. A paper sketch is cheap and easy to change. Html designs are still relatively simple to modify (or throw out). That's not true of programming. Designing first keeps you flexible. Programming first fences you in and sets you up for additional costs.
>
> Another reason to design first is that the interface is your product. What people see is what you're selling. If you just slap an interface on at the end, the gaps will show.[6]

Your product is *not* the CMS, and your product is not the back-end custom coding that you will do to make it work—it is the front end of your site that users interact with. In your customers' eyes, your product is the interface they see when they visit the digital branch. Obviously, design is all about what makes your site unique. There are limitless possibilities, but it is crucial that your design is attractive, well-organized, clear, and easy to use.

By starting your branch through design and sketching, you can organize yourself and be better prepared for programming.

The Design Process

When designing any website, it's best to start with the site's main page. Start by figuring out what content you need on the page and where you need that page to link. What services are important enough to be mentioned on that main page? Do you have a logo? Where should it be positioned on your page?

Next, you should get a piece of paper and start sketching ideas. It might be helpful to take a peek at other websites. Make sure to look at a few library websites that you respect, and see what they've displayed on their main page. But more important, get out your list of websites that your customers are using and examine those websites. Visit some universally visited websites, like Amazon, eBay, and CNN.com. Incorporate elements of those websites into your paper prototype.

Then start sharing your design ideas. If you have a Web committee tasked with redesigning your site, meet with them. Sketch design ideas together and share, then finalize the paper design by combining the best overall features and design ideas of each.

HTML Prototype

Once you have decided on your design, the next step is to do a mock-up that can reside on the Web. While the mock-up doesn't have to be fully functional, it should have some of the functions so you can get an idea of what it is like to click through the design. At this point, you're not really coding for functionality or using a CMS. Instead, you are making HTML representations of your paper sketches. Get this functioning at a very minimal level, then start showing it and gathering feedback.

Seeking Feedback

You should seek feedback from people who are going to interact with the site. I'd start with administrators and managers, as they are the staff members whose approval will be so important to launching the site, and in most cases they get ultimate veto power. You want to make sure they are on board from the beginning. Make sure this group likes the design and the direction you're taking.

Also check with staff, since they work directly with patrons. If you plan on doing usability testing, this is the time to do it!

During this process, expect a variety of comments. Don't take them all to heart, but rather listen for trends. For example, if one to three people say, "Those colors are ugly," but others like the colors, let those comments slide. On the other hand, if you hear that comment from half of the people that you talk to, you may want to consider selecting new colors.

I'd also suggest getting some feedback from customers. Library managers need to like the new design, and front-line staff need to help patrons with the new site. Ultimately, however, your patrons are your most important group. They are your library's community and will be the digital branch's most important customers. Ask them if they like the design, what they would change about it, and what they are expecting from a digital branch.

Once you have a good amount of feedback, start winnowing it. If certain things keep popping out at you, fix those issues. Add any requests that make good sense or new thoughts that came from the feedback. Show the revised version of the website around one more time, and do the same process again. Once that's done, you can start building.

Building: Getting Started

Now that you have the design roughed out, and you have buy-in from a number of people—both staff and customers—you are ready to start putting your site together. The most important things you'll need to get it built are time, staff, deadlines, and content.

Time

Time is first on this list for a reason. You have to set aside a lot of time to build a website. A simple blog can be built pretty easily—set up a free account, pick a name for it (the hardest part about blogging, besides the content creation), and choose a skin (or design) for your fledgling blog. Presto! You've built a blog.

A fully functioning library website is a different animal. A library website tends to be a fairly large and somewhat complex site that consists of many pages and a lot of content. There's the main page to design, but there will be other page templates to design and build, plus potentially thousands of individual pages that need to be copied over from the previous website to the new digital branch. Library websites are complicated and time-consuming to build.

It's important that you are realistic about the time it is going to take to build the site and the time that staff members will have to work on it. The day-to-day functions of the library need to continue while you are building your digital branch, and building can be a complex process. By using conservative estimates, you can avoid making promises that you can't deliver on quickly or giving out launch dates that need to be pushed back.

Deadlines

You will need to set some firm deadlines. No one ever builds the perfect website, but at some point, you have to release it, or go live with what you have. Remember, since this is a digital building, you can knock it down and start over, or add a room, relatively easily. It doesn't cost any actual money to do that unless you have hired a Web design firm to build your site, which is out of reach financially for many libraries.

Here's what Tom Mochal, President of TenStep, Inc. and columnist at TechRepublic.com, says about deadlines:

> I'm convinced that a project team without a firm deadline will be unfocused and will ultimately take much longer to deliver than necessary. It sounds like you have a similar concern. So, when I see projects that are like this, I work with the project team to have them set a reasonable deadline, and then I hold them accountable for that date. This allows the team to work with purpose and focus. It's also a way of making sure that the projects don't continue indefinitely. Even if the client doesn't have a sense of urgency on when the project is completed, I want to make sure that the project team doesn't have this same attitude.[7]

In order to keep your project on track, you'll want to set deadlines, even if the library board or other governing body hasn't set one for you. Otherwise, the staff building the project will be unfocused, might tend towards perfectionism, and won't be able to give time to other important tasks and projects.

How do you set deadlines and manage time? Easy. Ask your Web-building staff how much time they will need, and then add some extra "wiggle room" time to that estimate. Here's what Merlin Mann of 43 Folders suggests:

> In my days as a project manager (and in another life as a freelance designer), I got into a habit that has served me well to this day: get the best estimate of both job requirements and time-to-completion that you can find. Then add 20%. Then, when nobody is looking, add *another 20%*. Then pray.[8]

Staff

Of course, you have to consider the time of the people who will actually be building the site. You have basically two options here: use the Web staff you already have, or hire a freelance designer or Web design firm to build your site.

Obviously, we are all subject to financial constraints when it comes to IT staff. That being said, if you really want your digital branch to be an important part of your library, the best work will usually come from hiring a developer or dedicated staff member to build and maintain the site. If this is not possible, it is best to centralize these responsibilities as much as possible and keep them in the hands of a small number of staff members.

Once your site is built or redesigned, you will still need that Web staff onsite—there will always be a lot of work to be done. They will need to run analytics reports (and actually understand what those numbers mean). They'll need to fix things that go wrong. They need to train staff on how to post to the site and how to hack up a bit of HTML to make their posts better, and, of course, they will continue building new projects. Staff should be able to test the site, answer customer e-mails, help customers find information, listen to the hang-ups people are having with the site, and do usability testing. Once your site is "done" and has gone live, staff can have a sip or two of coffee and relax—for a week or so. Then they need to start planning the next release. You want to continually redesign, rather than doing it once every three years.

If you are hiring Web staff, make sure to hire people who know what they're doing! Yes, this sounds a bit obvious—of course you would hire someone who knows how to build a website for the webmaster job, right? Well, lots of libraries don't actually do this, believe it or not.

For instance, imagine a library hasn't hired a webmaster for nine or ten years. Now that person has moved on, and it's time to hire again. So what do some libraries do? They dust off that webmaster job description from 1998 and send it out again—not the best strategy when Web technology changes as fast as it does. Most likely, that job description mentions HTML and maybe (if you're lucky) CSS. It's old and outdated, and the only person who realizes that has left the building, so to speak!

So what can you do if you're in that situation, or one similar to it?

- If you ever have an IT staff member leave, ask the person who is moving on to update his or her job description before leaving, paying special attention to job requirements. This way, you'll at least get someone who has the skills to do what the previous person did.
- Ask another library to look over the job description. Someone recently had me look over their job description for an open Web staff position. What I described above pretty much held true. They had an outdated description that didn't include the skills they actually needed. With an update, they got applicants (they hadn't gotten any with the old job ad).
- Look at other job ads and the requirements they list. Don't look only at library job descriptions. Instead, go to places like the 37signals Job Board (see figure 12) or CrunchBoard Job Board and look at the job descriptions for Web developer positions. You can find helpful wording and job titles from those job sites.
- Talk to other people in your community who have hired IT staff. If you have friends or know of local businesses that hired people, talk to them. They can help you understand the process and might even be able to give you some contacts.

37signals Job Board
http://jobs.37signals.com/jobs

CrunchBoard Job Board
www.crunchboard.com/jobs

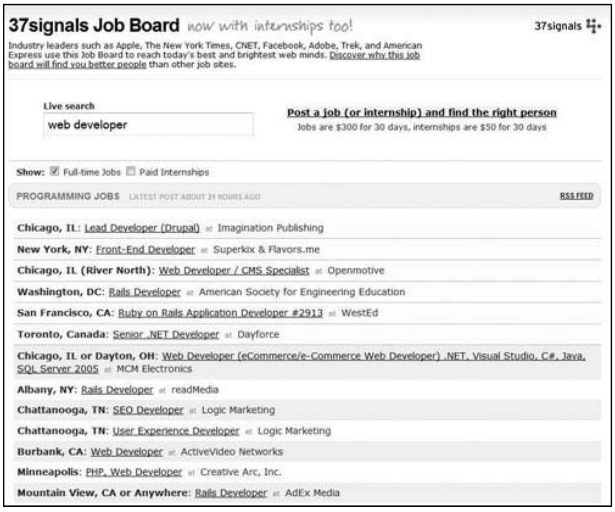

Figure 12
The 37signals Job Board.

Creating Content

Finally, you need to start thinking about content. You can't actually go live with the site if there's no content on it! That would be like opening a new branch library without ever thinking about what collection would be housed in the building. One way to create content is to start adding content to the new site one to two months before the digital branch is ready to go live, even if no one sees it. Think of this as buying and shelving the books in preparation of a new branch library opening, only digital!

What types of content do you need? That depends on the goals you have for your digital branch. Here are some of the usual components:

- *Traditional content*—Think of this as being like an About Us page, a history of the library, policies, information on the library board, addresses, directions, and hours.
- *Catalog*—You're a library. You need to have your library catalog online and linked from your site. Believe it or not, not all libraries have these yet, and some don't have their catalog linked on their website yet.
- *Databases*—You pay an arm and a leg for these things, so you need to link them, describe them, have online tutorials about them, and get them in front of people's faces!
- *Links*—You will most likely have a reference department or a public services staff member who is good at finding things online and wants to remember where some of those things live, so he or she can point them out to customers when needed. You'll need to have a place for those useful Web links.

Then, there's all the 2.0 stuff. Web 2.0 has brought libraries and websites a very long way. We'll talk more about the whole digital community aspect in a later chapter. But for now, what types of Web 2.0 things should you be thinking about? Here are some ideas:

- *Blogs*—You will want a blog—at least one, with comments. Make sure the blog talks about what's new at the library—new books, upcoming events, and anything exciting going on at your library.
- *Flickr*—You should be taking pictures at your library—pictures of your library, pictures of cool people doing cool things at your library. Put those pictures in Flickr! Flickr allows you to store photos and include them on your website, and it allows people to comment on the photos. You can also subscribe to a photo feed. Photos are a great way to convey the visual aspect of your library.
- *Videos*—Flickr, YouTube, blip.tv, Vimeo—all are great places to upload videos and then to embed them on your website. Take video snippets of events and use them as visual moving ads for the next similar event. Or use them as a "this is what happened" video, to keep people's interest after the event happened. You could also use videos to do an online book review.

If I summed this chapter up in one sentence, it would be this: think, plan, and do. First, think about what it is you want to achieve with your website. Then do a lot of planning to meet those goals you dreamed up. And finally, do it—build the site you planned!

And then do it all over again.

Notes

1. "What Is a Content Management System, or CMS?" Enterprise Content Management, www.contentmanager.eu.com/history.htm (accessed Feb. 2, 2009).
2. "Drupal," Wikipedia, http://en.wikipedia.org/wiki/Drupal (accessed Feb. 6, 2009).
3. Joomla! Community Showcase, http://community.joomla.org/showcase (accessed June 20, 2009).
4. "ExpressionEngine Products and Services," Expression Engine Store, https://secure.expressionengine.com/index.php?ACT=EE (accessed June 13, 2009).
5. "Showcase," ExpressionEngine website, http://expressionengine.com/showcase (accessed June 13, 2009).
6. 37signals, "Interface First," *Getting Real: The Smarter, Faster, Easier Way to Build a Successful Web Application* (Chicago: 37 signals, 2006), http://gettingreal.37signals.com/ch09_Interface_First.php (accessed Feb. 6, 2009).
7. Tom Mochal, "Projects without Firm Deadlines Can Lack Focus," TechRepublic, June 25, 2003, http://articles.techrepublic.com.com/5100-10878_11-5035134.html (accessed Feb. 7, 2009).
8. Merlin Mann, "Task Times, the Planning Fallacy, and a Magical 20%," 43 Folders, Aug. 13, 2008, www.43folders.com/2008/08/13/estimating-time (accessed April 29, 2009).