Nina McHale

Steal this code! Please!

Creating HTML widget generators for libraries

The embedded, widget-ized library is nothing new; there are iPhone library and catalog applications, vendor-created chat widgets, and iGoogle gadgets galore. Montana State University library users can drop an iGoogle gadget into their iGoogle homepages that provides quick access to

Fig. 1. QuestionPoint's Qwidget, Montana State University iGoogle Gadget, and WorldCat Local iPhone app. Visit this article online for detailed images.

"Books & More," "Quick Articles," their "Journalist," and "Google Scholar" in a tabbed interface. Reference librarians who use OCLC's QuestionPoint can embed a Qwidget, a chat widget, easily into library Web pages. iPhone and iPod Touch users have libraries around the world in their pockets and purses with the WorldCat Local app.

These helpful tools offer libraries a means to insert library tools and content into venues that are outside of the traditional domain of library Web sites. Additionally, many database vendors are now offering code editing tools to enable users to roll their own widgets with their database offerings.[1]

These are just a handful of innovative widget examples from libraries and vendors. These creations, while ingenious, are nonetheless limited by their platforms, some of which are costly. iGoogle gadgets, while free, are for people who use or are willing to try iGoogle homepages. iPhone apps are for iPhones and iPod Touches, and the Qwidget requires a library license with OCLC's QuestionPoint. Can we take this concept a step further and share the love by creating our own HTML widget generators?

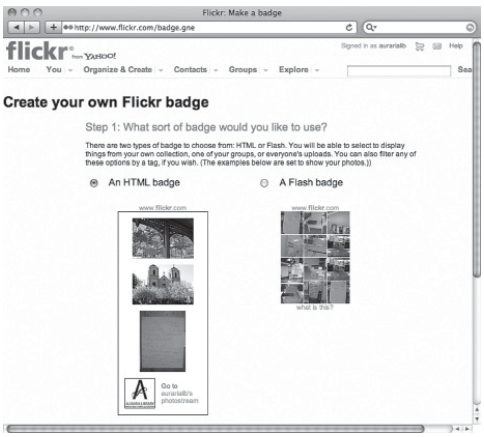

Fig. 2. Flickr Badge Generator.

For example, consider the Flickr badge creator (www.flickr.com/badge.gne). The badge creator is a tool that provides Flickr users a means to create and edit code to insert HTML or Flash code snippets into any Web pages to which the user is able to insert raw code. The result is a stylish rotating display

Nina McHale is Web librarian at University of Colorado-Denver, e-mail: nina.mchale@ucdenver.edu

of any set of one's Flickr photos. It is, essentially, a widget generator. Regardless of HTML or programming skill level, academic librarians can create their own such code generating tools to offer teaching faculty the ability to leverage the search power of library tools; these code snippets can be "stolen" by teaching faculty and inserted into course management systems like BlackBoard, WebCT, and eCollege, and personal Web pages and Facebook pages—any place that people can cut and paste raw HTML code. To take provision of this kind of service to this next, platform-free level, all it takes is the ability to create a simple Web page, analyze a bit of HTML, and cut and paste.

In technical terms, at the simplest level, any library search box or fill-able box on a Web page is an HTML form. While writing form code from scratch is beyond basic HTML knowledge, copying and reusing existing form code is as easy as Ctrl + C and Ctrl + V (or Cmd + C and Cmd + V for Mac users). Consider the basic code for the Auraria Library Skyline catalog's search box:

```
<form action="http://skyline.cudenver.edu/search~/" method="post">
   Find a Book at the Auraria Library:
         <select name="searchtype2">
                  <option value="t">Title</option>
                  <option value="a">Author</option>
                  <option value="X">Keyword</option>
                  <option value="d">Subject</option>
         </select>

   <input type="text" size="17" maxlength="75" name="searcharg2" />
   <input type="hidden" name="SORT2" value="D" />
   <input name="Submit" value="go" type="submit" />
</form>
```

Fig. 3. HTML Form Code Example for Skyline Catalog Search Box.

These dozen lines of code yield the following in a Web browser:

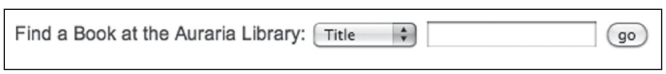

Fig. 4. Skyline search box.

Believe it or not, that's all the HTML it takes to create a simple catalog search form with a dropdown for search type—that is, if you are an Innovative Interfaces, Inc. (III) customer. Which brings me to my caveat: the code examples shown here are for widgets for specific, existing, products and services that Auraria Library uses or has used. These include our III catalog and Serials Solutions 360 Search (recently cancelled, yet included here as an example nonetheless). Readers may freely use any of the code from any of these examples, but the examples will need editing to suit local installations of products and services (i.e., the 360 Search code won't work without a current subscription to 360 Search). If necessary, seek guidance from in-house Web/IT people for help editing these examples or identifying the code bits that drive your library's tools and services.

So, let's get started! First, create a Web page—or a set of pages, if more than three or so widget generators are desired—in which to offer widget generators to your

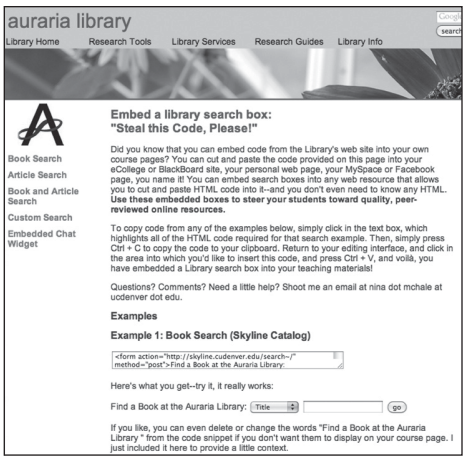

Fig. 5. Auraria Library Steal This Code! Page.

public. Write a welcoming paragraph or two that describes the purpose of the widget generators, and offer contact information for assistance in creating widgets (see figure 5). Next, consider which widgets you would like to offer. Generally, search widgets are useful and practical, and widgets for services like IM chat can popularize and market existing services.

Next, identify the HTML code necessary to "drive" these widgets. Keep in mind that for some applications, the code might be

overly complicated for packaging into a widget generator. (This was the case when Auraria switched from the 360 Search federated search product to WorldCat Local's NextGen catalog solution; recreating the search box that now appears on the library's home-page is too code intensive to replicate with the method described below.) Viewing the source in a Web browser and searching for <form> tags can help isolate just the bits of code needed.

```
<textarea name="skyline" cols="50" onfocus="select();">
<form action="http://skyline.cudenver.edu/search~/" method="post">
   Find a Book at the Auraria Library:
        <select name="searchtype2">
                <option value="t">Title</option>
                <option value="a">Author</option>
                <option value="X">Keyword</option>
                <option value="d">Subject</option>
        </select>

     <input type="text" size="17" maxlength="75" name="searcharg2" />
     <input type="hidden" name="SORT2" value="D" />
      <input name="Submit" value="go" type="submit" />
   </form>
</textarea>
```

Fig. 6. HTML Form Code Example for Skyline Catalog Search Box

Now it's time to create a generator. Each chunk of form code will need to be wrapped in a pair of HTML <textarea> tags in order to allow your users to easily copy and paste all of the code into their own Web pages.

In figure 6, the opening <textarea> tag provides a name for the textarea, simply "skyline" in this case. Give each textarea a descriptive and unique name for the tool that's being used. The next part of the code—cols="50"—determines the width of your textarea with the HTML cols attribute. Increase the number if you'd like it to be wider; however, reducing the value to much less than 50 might result in an oddly skinny widget generator box. The final part of the opening <textarea> tag—onfocus="select();"—is the most important part. Including this little bit of JavaScript causes all of the code within the textarea to be highlighted when the textarea is clicked. Don't forget the closing </textarea> tag at the end of the code chunk. Along with the code, show a live, working example of what the code will produce (see Figure 5).

Will teaching faculty be able to use it? Figure 7 is an actual example by Department of Education University of Colorado-Denver faculty member Jenna Ream, who inserted a 360 Search box that allowed her students to search two education databases simultaneously. Because Serials Solutions provides excellent documentation for creating 360 Search search boxes with any combination of federated databases, including sample form code for the search box, creating custom widgets of bundled databases for any subject area was simple. Check with your catalog, database, and search solution providers to see if they

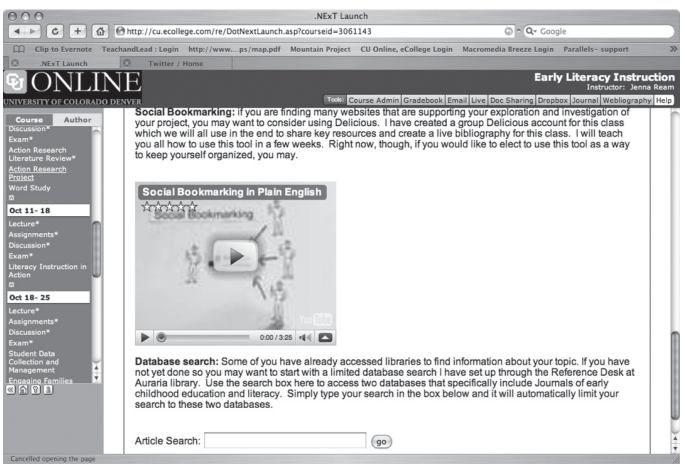

Fig. 7. University of Colorado-Denver e-College Early Literacy Instruction course page by assistant professor Jennifer Ream.

provide this level of coding support. Figure 8 shows the form code that Ream inserted into her eCollege course page to allow her students to search two education databases—WilsonWeb's Education Full Text from and Education from Sage—to allow her students to search library resources from inside their course page. The Serials Solutions code allows customers to change which databases are searched simply by changing the three-letter codes in the <input> tags.

```
<form action="http://0-TB4CZ3EN3E.cs.serialssolutions.com.skyline.cudenver.edu/resultFrameset.jsp"
name="searchForm" method="post" target="new">

    <input value="TB4CZ3EN3E" name="SS_LibHash" type="hidden" />
    <input value="default" name="catGroupList" type="hidden" />
    <input name="searchBy" type="hidden" value="Category" />
    <input value="title" name="field" type="hidden" />
            <input type="hidden" name="dbID" value="WEB" /> <!--Education Full Text-->
            <input type="hidden" name="dbID" value="SED" /> <!--Education Sage-->

    <div class="SSCentralSearchSearchTerm">
    <span class="SSCentralSearchSearchCriteria">Search the Auraria Library for Books and Articles:
    <input class="SSCentralSearchSearchCriteria" maxlength="1000" size="25" name="term"
    type="text" value="" />
    </span>
    <span class="SSCentralSearchSearchTermSubmit">
    <input class="SSCentralSearchSearchTermSubmit" value="go" type="submit" />
    </span>
   </div>
</form>
```

Fig. 8. HTML Form Code Example for Serials Solutions' 360 Search, federating Education Full Text and Education (Sage) Databases.

Keep the copy-and-paste process simple, with easy-to-read directions, and make contact information for an appropriate person readily available. Further, consider demonstrating how to copy and paste the code successfully into your institution's course management systems of choice at faculty outreach events. In fact, the iteration of the Auraria Library "Steal This Code!" page presented here came from a spring symposium conference presentation on the University of Colorado-Denver campus.

Another possible concern is security. You may think, "Aren't we exposing ourselves by showing everyone our code?" If the code used in the chat widgets is already in use on a public-facing Web site, the widgets are not giving anyone access to anything that they could not already see by viewing the page source in a Web browser. Authentication procedures are not circumvented by access to the raw code; a proxy server will still stop anyone who is not authorized to use protected resources, such as subscription databases, regardless of where the widgets are placed. The greatest risk is that of success—that offering code for these kinds of widgets would increase usage, which is, of course, a good thing.

There are certainly higher tech means of creating this type of resource. Two excellent examples are the University of Minnesota- Duluth Library and the University of Washington Libraries. With a bit of local programming savvy and creativity, the University of Minnesota-Duluth is able to offer widgets for catalog searches, e-journals, library links, and an assignment calculator. The code copying process is made even easier by a "get code" button, which is generated by a dozen or so lines of JavaScript code. The University of Washington Libraries use Widgetbox, a free online widget creator, to offer a widget creator that leverages into widget form their instance of WorldCat Local.

If the two examples above are beyond your skill level, never fear; by following the steps in this article and creating a simple "Steal This Code!" Web page, academic librarians can provide platform-independent widget generators that teaching faculty can use to inject library content directly into their online course materials. Creating a widget-generating resource by using the HTML <textarea> tag is a low-tech, easy, and fun way to extend the reach of library resources beyond the environments that we control.

Notes

1. Major vendors offering databases packaged as widgets include EBSCO (www.widgetbox.com/tag/ebsco), ProQuest (www.proquest.com/en-US/utilities/widgets/index.shtml), and Gale Cengage (access.gale.com/widgets/).