Slide 1:
Learn techniques for optimizing reusable content while creating flexible designs in LibGuides. This presentation will focus on using widgets and flexbox styling with LibGuides to reduce updating workflow and improve responsive design.
20 minute presentation (5-10 min for Q&A)
Slide 2:
Hi, I'm Terri Holtze. I'm the Head of Web Services for the University of Louisville Libraries.

The presentation slides and my notes are available online at this address – so you don't have to worry about taking notes.

First, a little background…
Slide 3:
We are in the midst of a major transition with our website. We've been using LibGuides for database management as well as subject and course guides since 2009. The rest of our site has been in the University's content management system (CMS), Plone. We are in the process of moving our entire site into LibGuides CMS.

So during this transition I've had some time to think about architecture of our site and one of my major goals was to make effective use of reusable content. Reusable content – content that is linked to a master version and updates automatically if the master is updated – is one of the strengths of the LibGuides system and has two main benefits:

It reduces workload, and
It reduces the number of bad links (links on pages we might've forgotten about)
Slide 4:
None
Slide 5:
LibGuides provides some easy-to-use mechanisms for reusing content including:
A-Z list: provides a stable environment for keeping current information for your databases.

Links: While you can add links to any guide or add them through the asset manager, I like to add links to our Site Index guide and then use that as the official link for internal resources. When you add a link and choose the reuse a link tab, you can search for the link and the screen will show the owner of the Asset and the guide it was created on. This gives librarians an easy way to identify which version is the official copy.

The biggest problem for reusability is that links created in a Rich Text Box are not reused links. They will not update automatically from a master link.

Boxes: Just like links, when you go to add a box on a page you have the option of reusing a box (from another page, from another guide). This is how you'll reuse boxes for most of your LibGuides, but sometimes you need more flexibility.
Slide 6:
LibGuides was designed to reuse content. In the Asset Manager and in the A –Z list you can view the mapping count which tells you how many places the asset is used and if you click on the number it will bring up a list with links to each of the guides reusing the content.

Slide 7:
The CSS files you use are reusable and while technically they are code for design and layout rather than content in their own right I thought I'd throw in this tip.

Rather than adding CSS in the guide's custom css, I highly recommend adding css files as linked customization files. In the Admin toolbar choose Look & Feel and then look for the Upload Customization Files section (which is a collapsed section of the page so it can be tricky to find the first time)

This allows you to use the style sheets for multiple pages/groups, AND
You're less likely to crash the guide. Really bad code can make the guide un-editable forcing you to rebuild it. If the css and script files are linked you can temporarily take the linked file down and your content will be editable again.

When you upload them as customization files - LibGuides even gives you the exact code to add into the guide's (or the group's) Custom JS/CSS.
Slide 8:
None
Slide 9:
Admin – Tools - Widgets

Under the Tools button in the LibGuides admin bar you have the choice to create some pre-set Widgets.
If you haven't explored this you really need to – there are a lot of great things you can do.

For example, some of the things I've used it for include:
Slide 10:
creating a list of just the course guides of a subject Anthropology guides - by using the Guides tab, and using the filter options for Subject and Guide Type (Course Guide)

OR
Slide 11:
Using tags to identify certain types of content and then creating widget that filters to guides with that tag.
So here the list of forms is actually a widget that searches the system for all guides tagged as "Forms" to generate this list on the fly.
Slide 12:
Once you've filtered the results to what you want you can copy the embed code you'll need.
Slide 13:
The code was originally intended to help you embed something from your LibGuides system into another content management system.

The code includes all the scripts and stylesheets needed to replicate your LibGuides design in that new location, but if you try to include all those files again inside LG, **bad things happen** – like the box not displaying correctly or, more importantly, no longer being able to edit the guide.
Slide 14:
You need to remove any system-level code and just leave the pieces needed to generate the box

content.
You want to look for where the last stylesheet ends. Choose that stylesheet and everything above it and delete them.
The remaining code is what you'll paste into a media/widget section or, alternatively, into the Source view of a rich text section in your LibGuides box – which I'll talk about in a minute.
Slide 15:
So what you have left is this:
Script for tracking the site usage
Scripts for displaying pulling and displaying the box
And a div with a unique identifier where the script places the content

And once you have one embedded element on your page you can eliminate the top script as well for subsequent embedded widgets.
Slide 16:
Next I'm going to talk about embedding widgets within rich text – which is a technique I use to improve the design flexibility and security of a guide.

Let me give you an example.
Slide 17:
This is the homepage for Archives.

A number of our archivists keep the content up to date on this section of the site. I wanted them to be able to update the content as needed without giving them permissions to delete the guide or change the layout. So they don't actually have edit permissions on the home, but they have edit permissions to change the content of the boxes that actually reside in another guide.

"Research in the Archives" is a simple reused box from a separate guide that appointed archivists can edit.

They can also edit the links that appear in the dropdowns, but this takes a more advanced method – embedding the widgets within the Bootstrap dropdown menu HTML. I could've created these dropdowns just as straight HTML, but if I had the links couldn't be automatically updated. By creating these as a box with a list of Links I could embed them into the dropdown HTML. This allows the archivists to edit the content of the dropdowns without touching the homepage and as links are updated they'll automatically update on the homepage as well.
Slide 18:
Tokens are another way of working with reusable content, but they only work within the Templates.
Admin – Look & Feel – Page Layout – Customize Page Layout

Access to this area requires Administrator level permissions.

Wherever you see the double curly brackets this is code pulling in content from each guide's metadata when that guide loads. They're called tokens and there are a number of them you can use to customize the layout of information in your guides.

[Instructions for editing templates: http://support.springshare.com/libguides/guidetemplates/

customizeguidetemplates]
Slide 19:
Token list with descriptions at: http://support.springshare.com/libguides/guidetemplates/customizeguidetemplates#s-lg-box-3821

So there are a number of elements you can reuse or customize the location of in your templates.
Content boxes
Page and guide title
Header and footer information,
Etc.
Slide 20:

Another thing you can do in the template is inject a little code to pull customized, content-specific help information.

On our side tab here, the question mark brings up a pop-up screen populated based on a tag entered into the guide. In this case the tag is Josh's name. The script then finds the box on the guide titled "Josh Whitacre" and shows it in the pop-up.
Slide 21:
This code is included in the guide page layout template. It and the CSS code tell the browser where to display the pop-up box.
Slide 22:
Along with this code which tells the browser where to display the pop-up link that appears on the side tab.

And a script that appears just before the </body> tag of the template.

In order to make the script work more quickly we created a group called Help Modals to store the guides with the contact info. That means the system only has to check the guides in that group to find a match – significantly speeding up the process.
Slide 23:
So basically, there are 4 elements to make this happen:
A guide for the specific individual with their contact info as the only box and their name in the tag field
Additional HTML in the guide template to provide a location for the pop-up link and the pop-up box.
The script (also in the template)
CSS telling the browser where the pop-up link and the pop-up box should appear on the screen.
Slide 24:
I'll be happy to take any questions.