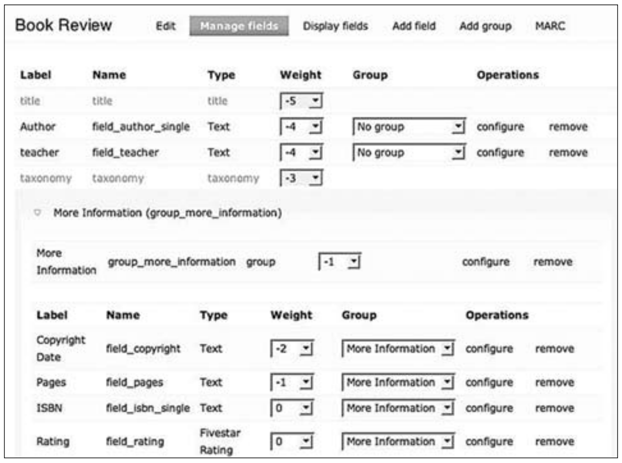
The Trifecta

Assuming we haven't already lost all credibility by repeatedly claiming that developing custom sites within the Drupal framework doesn't require code and is actually quite easy, let us now reveal the secret. The majority of our work really does not involve custom code, but rather makes use of three contributed modules that combine to create a point-and-click application-development environment. The Content Construction Kit (CCK), Views, and Panels modules work together to make this possible.

CCK

The Content Construction Kit module is really a framework within a framework to extend Drupal's node system by allowing the creation of custom content types. For example, a library could create a new custom content type named Purchase Request. Site visitors could then be invited to submit a request for a new book they would like the library to consider purchasing. CCK starts with the basic title and body, but also allows new fields to be added (see figure 16).

A purchase request module would probably best be set up to have the desired item being stored as the node title. To clarify this for users, however, the usual label "Title:" can be changed to read "Title of item being requested:" You can then ask for additional information about the item: a text field could collect the author, a link field could ask for a link to a Web site showing the item, and an e-mail field would let the library contact the user about the request. To maintain security, you can collect this information without ever displaying it back anywhere on the public Web site. If there are concerns about abuse of the form, it can be restricted to access by Authenticated users.



Label	Name	Type	Weight	Group	Operations	
title	title	title	-5			
Author	field_author_single	Text	-4	No group	configure	remove
teacher	field_teacher	Text	-4	No group	configure	remove
taxonomy	taxonomy	taxonomy	-3			

More Information (group_more_information)

More Information	group_more_information	group	-1		configure	remove

Label	Name	Type	Weight	Group	Operations	
Copyright Date	field_copyright	Text	-2	More Information	configure	remove
Pages	field_pages	Text	-1	More Information	configure	remove
ISBN	field_isbn_single	Text	0	More Information	configure	remove
Rating	field_rating	Fivestar Rating	0	More Information	configure	remove

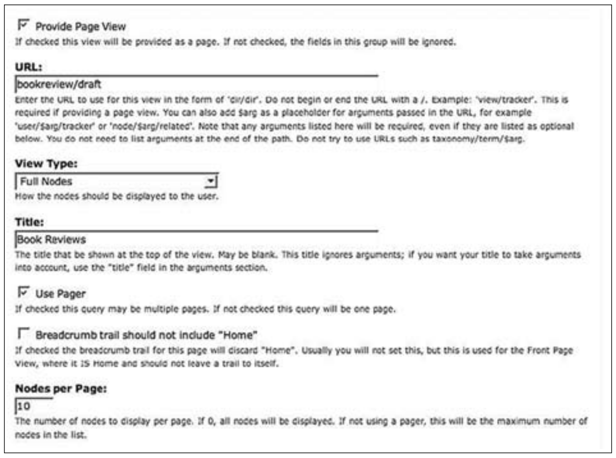
Figure 16
Using Drupal's Content Construction Kit module.

It must be stressed that it would be very difficult to overstate the power and potential for using CCK. Through the very simple graphical interface of the module, you are actually creating a Web form that collects data and deposits it in a database. In the past, this has been an incredibly complicated action, often requiring the very expensive time of a Web application developer. When we began looking at offering book reviews for student use, we had to make code changes to a special Book Review module. Now, with CCK, we have been able to add a BCCKreview module to Fish4Info that is more powerful and much easier to use. A CCK custom node is also at the heart of the Fish4Info catalog; it provides full MARC integration with CCK fields for MARC fields we want to display.

Views

Once data has been collected through CCK, the next task is to display the information back to users. To return to the example of a purchase request, you will need to create a report that lets librarians see incoming requests. Though in the past this would have required custom code, the Views module handles any reporting needs by working as an incredibly complex filter. If all of the nodes in Drupal exist in a giant pot of soup, then Views is a dynamically slotted spoon that can instantly capture any type of content.

Views are built from a number of elements: the output display type, the fields to be displayed, the filters to be applied, and the sorting to use. Views can be created as pages and/or blocks in a variety of styles ranging from a display of the full node to a custom list of fields in a table. Table and list views can be built to include only selected fields from the node (see figure 17).


[✓] Provide Page View
If checked this view will be provided as a page. If not checked, the fields in this group will be ignored.

URL:
bookreview/draft
Enter the URL to use for this view in the form of 'dir/dir'. Do not begin or end the URL with a /. Example: 'view/tracker'. This is required if providing a page view. You can also add $arg as a placeholder for arguments passed in the URL, for example 'user/$arg/tracker' or 'node/$arg/related'. Note that any arguments listed here will be required, even if they are listed as optional below. You do not need to list arguments at the end of the path. Do not try to use URLs such as taxonomy/term/$arg.

View Type:
Full Nodes
How the nodes should be displayed to the user.

Title:
Book Reviews
The title that be shown at the top of the view. May be blank. This title ignores arguments; if you want your title to take arguments into account, use the "title" field in the arguments section.

[✓] Use Pager
If checked this query may be multiple pages. If not checked this query will be one page.

[ ] Breadcrumb trail should not include "Home"
If checked the breadcrumb trail for this page will discard "Home". Usually you will not set this, but this is used for the Front Page View, where it IS Home and should not leave a trail to itself.

Nodes per Page:
10
The number of nodes to display per page. If 0, all nodes will be displayed. If not using a pager, this will be the maximum number of nodes in the list.

Figure 17
Using Drupal's Views module.

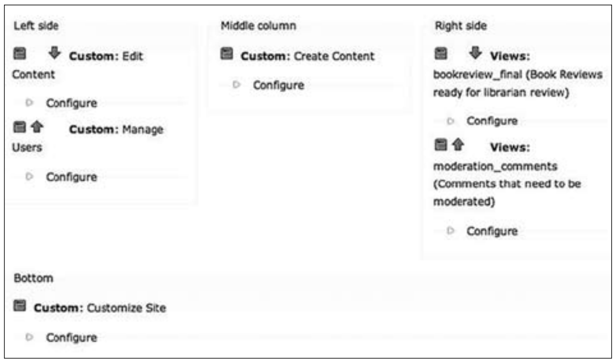
To display purchase requests, you may want to create a view that will display as both a block and a page. When a librarian is logged in, you might want to display a sidebar block listing the titles of the last five purchase requests sorted by the date submitted. We could also use a filter to make sure that the request hadn't already been approved or denied. This would give a quick overview of new requests without using much screen space while also allowing the librarian to click on a title for the full node. To see more than the last five requests, the librarian could go to the page-style view, which could present the requests as teasers or even full nodes in a list. Using exposed filters would also allow the librarian to further limit the view to see requests from a certain user or requests that would go to a certain department.

Panels

Now that you have a page and/or block of purchase requests, you need to be able to display it on a page in a way that won't overwhelm librarians. While blocks and block regions can help, they don't provide organization within the main content area. This is where the Panels module comes into play. Panels are preformatted display templates for the main content area of a Drupal site. The panel templates included with the module are named by the number of columns included and whether or not they are stacked between a full-width header and footer. Panels are further identified by the percentage widths of the columns. For example, a 50/50 panel has two columns, each of them half of the content width, while a 33/34/33 panel has three roughly equal width columns.

Panels are an incredibly flexible way to lay out a display. Any of the panel regions can be filled with a variety of "stuff," including content/nodes, views, blocks, or custom text/HTML. Including a node is especially useful for highlighting a permanent welcome message or other front-page content. Remember, though, that nodes do not have to be text. You can use images in a node to create a graphical header to your content area. If you want to include dynamic content, such as the latest blog post from the reference desk or the five newest books, you need to use a view in your panel. You can build a "Refdesk" view that filters for posts only to the reference desk blog, sorts them so the newest is first, and then displays just one post on the page. Now by including that page in your panel, you have a dynamic display of the latest post placed in a static location. This is different from just promoting to a generic front page where the post would be displayed in a list of all the other promoted content. Narrow columns are ideal for blocks that might normally be displayed in a sidebar. Since the panel is part of the main content area display theme, blocks displayed here appear to flow together with the rest of the content, as opposed to being set apart in a sidebar. If none of these meet your needs, you can also include a custom HTML area to add anything from simple text to a Flickr widget (see figure 18).

Returning to the example of purchase requests, let us explore how the view you created might fit within a larger panel schema. One training issue we found while developing our Fish4Info portal was that the administration pages can be overwhelming to users. To compensate for this, we used a panel to create a librarian dashboard. The dashboard is a combination of custom HTML areas with links just to the management pages a librarian would need to use, as well as additional blocks and views. So a library that wanted to display purchase requests might want to include a view of the latest requests as part of a dashboard, along with other library updates. Panels make


Left side

Custom: Edit Content

Configure

Custom: Manage Users

Configure

Middle column

Custom: Create Content

Configure

Right side

Views: bookreview_final (Book Reviews ready for librarian review)

Configure

Views: moderation_comments (Comments that need to be moderated)

Configure

Bottom

Custom: Customize Site

Configure

Figure 18
Using Drupal's Panels module.

this not only possible, but very sustainable; once the view is established and placed in a panel, Drupal does all of the updating.

Putting It All Together

The Genesee Valley BOCES School Library System runs a subscription service for member libraries to provide media services. One of these services is a television taping service that allows users to request television programs to be recorded for classroom use. For obvious reasons, having the request system online is essential. We had originally developed an in-house solution programmed in Microsoft's ASP that had all of the maintenance and security issues of any homegrown system. Additionally, any changes in functionality required lots of custom coding. We needed something that integrated with our Web site, was reliable and secure, needed little maintenance, and was extensible as our needs changed. Of course, we used Drupal.

First, we downloaded and installed the following modules to supplement our Drupal core installation:

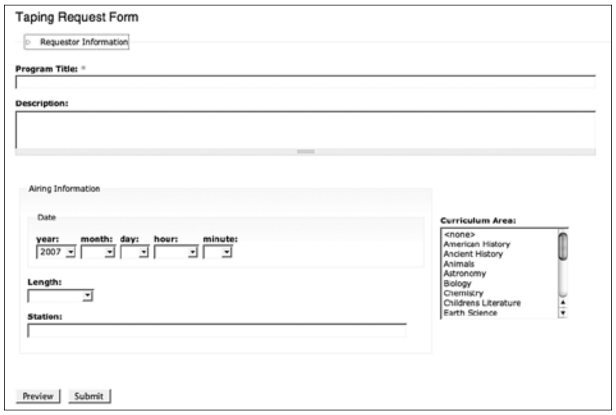
- CCK—allows the creation of custom content types
- DateAPI and Date modules—gives us a field for selecting date and time
- JS Calendar—provides a nice graphical calendar for selecting dates
- Email module—gives us a field for e-mail addresses
- Views—a module that lets us create custom lists of nodes

To get started, we first created a new content type, "Taping Request," using CCK. To the standard title and body, we also added fields for requester's name, school, e-mail address, program title, station, date, and time. By default, the node is not published; only certain roles have the ability to administer content and therefore publish these requests so that they are viewable to everyone. Anonymous users do not have the permission to publish these content types, so they sit and wait for approval from our media staff.

By just putting these pieces together and doing some configuration on the Administer Content Types pages, we had a Drupal form for site visitors to request tapings. After clicking on a link to the "Taping Request" page, the user is given the form to create a new taping request. Once the user submits the request, Drupal creates an unpublished node that is then displayed in a view. The view uses a filter to display all unpublished taping requests. These requests occur infrequently enough that we do not expect the staff responsible for tapings to constantly check the Web site for new requests. Instead we added some more functionality from additional modules to supplement the view.

- Subscription—allows users to subscribe to get e-mails when additions are made
- Calendar—a special view that allows users to look at nodes in a calendar format

The next step was to set up e-mail subscriptions for the media staff. The subscription module lets users get e-mails of new nodes that are added to the site. You can specify what kind of nodes prompt e-mails, as well as taxonomy terms or comments on individual nodes. Now, whenever a new node of the type Taping Request was submitted, the media staff received an e-mail with the details of the taping request and a link to the taping request. We also set up views listing the taping requests in chronological order so that they could always see the next taping requests as they came up. We set this up in a straight list view and in a calendar view so that they had the option of seeing the requests either way. Sometimes it is easier to glance at a calendar and see a graphical notification (the day is colored if there is a request) upon which to act (see figure 19)


Taping Request Form

Requestor Information

Program Title: *

Description:

Airing Information

Date

year: 2007 month: day: hour: minute:

Length:

Station:

Curriculum Area:
<none>
American History
Ancient History
Animals
Astronomy
Biology
Chemistry
Childrens Literature
Earth Science

Preview Submit

Figure 19
Taping Request Form page.

This quickly surpassed the functionality that we already had with our previous taping request system, which had taken considerable time to construct. With Drupal, however, we also have the flexibility to continue to reuse our content type. Oftentimes, the same television programs get requested by many teachers, or teachers might not notice good programs coming up. So we want to give teachers and librarians a way to not only request tapings but also promote upcoming programs and share programs that are already requested. To accomplish this, we created public views of the taping requests that included requests that were published and promoted by the media staff. Using the Signup module, users could sign up for a taping already on the site by simply adding their name, school, and e-mail address to the existing request. Using views, we can also provide an RSS feed of upcoming taping requests. If we wanted to, we could add categories for subject areas and provide feeds by subject area as well. Or we could use the Subscription module once again and give users e-mail messages when a new featured program is added in their subject area.

This example shows the real power of Drupal. As a content management framework, Drupal is uniquely suited to building Web applications from extensible modules in a friendly and easy-to-learn environment. There was no custom coding involved in this example. After dropping the modules into the correct directory on the Web server, everything was done through Drupal's graphical interface. The expertise demonstrated here is not knowledge of computer programming, but rather an awareness of the different modules available to supplement and extend the Drupal core functionality.