

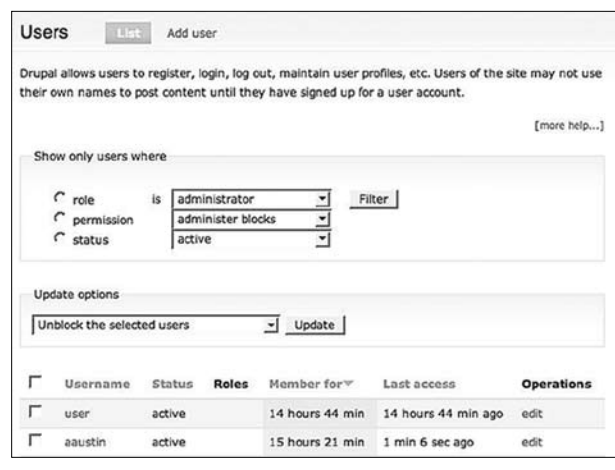
User Management

We previously mentioned that, rather than use the term *CMS*, many in the community prefer to call Drupal a *CMF*, or content management framework. Others prefer just to change the *C* from *content* to *community* or *collaboration*, and use the phrase *community management system*. This is to emphasize Drupal's superior user-management features that are especially suited to collaborative content authorship.

Using the Drupal Web interfaces, you can create users, identify user types or roles, and fine-tune permissions based on these roles. You can also empower users to register themselves on your site (with optional approval) and manage their own profiles, usernames, and passwords. To bolster the core user management, user management has been designed to be extensible. Many other modules have taken advantage of this and are available to expand core user management to address more specific needs (see figure 9).

At a basic level, the Drupal user-management system controls which users can see what content in your site. Perhaps more important, it mitigates the layers of permissions for contributors to the site, refereeing who can submit and edit what content and what happens to that content when it is submitted. Drupal does not dictate a specific work flow for Web authoring, but instead, with the right configuration, it can be adapted to fit the publishing work flow that is needed within your organization. This can allow you to distribute work on the Web site throughout a library. The Children's Services department head could be given permissions to create and publish new stories, while other staff in the department may be able to create content but not publish it without additional authorization.

Drupal handles these distinctions by using a role-based system of permissions. Users are assigned roles, and



Username	Status	Roles	Member for	Last access	Operations
user	active		14 hours 44 min	14 hours 44 min ago	edit
aaustin	active		15 hours 21 min	1 min 6 sec ago	edit

Figure 9
Drupal screen for managing users.

these roles are given specific permissions. Permissions can be assigned only to roles, not to individual users. Given this, it is important to understand that users can be assigned multiple roles and their permissions accumulated depending on which roles they have been assigned. This is different from many other systems, in which a user may be placed only in a single system role or group. Multiple roles can be created to build a scaffolding of permissions.

Using Roles

When you install Drupal, there are already two roles defined:

- Anonymous—users who have not logged in. Drupal treats all Anonymous users the same way; it has to because it doesn't know anything to differentiate them. This means that there is only one level in the Anonymous hierarchy.
- Authenticated—any user who has logged in. Users will be considered Authenticated users even if you assign them additional roles, as this is the top-level role in the Authenticated hierarchy. It is critical to remember that every user who logs in gets any permissions granted to Authenticated users (see figure 10).

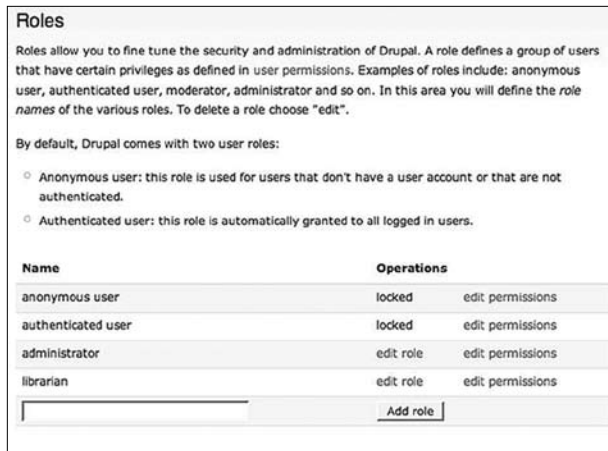


Figure 10
Drupal screen for managing roles.

During the installation process, you create a super-user account (with an internal ID of 1) that bypasses all security checks. This user does not need any roles or permissions assigned; instead, this person always has permission to do anything. Using this account is sort of like using root in Unix; you want to use it only when necessary, and you may even want to disable it for security reasons. Instead of using this first account, you will probably want to create a role called Administrator and have other users who will be administering the site assigned to this role. You will just need to assign all the permissions to the Administrator role and remember when you install new modules to revisit the Permissions table and give the Administrator role the new permissions that come with the new module.

Additional roles can be set up for different groups of staff in your library depending on your needs. Since our staff is small and our organization is not very complex, we generally add only three roles for our organizational sites: Administrator, Staff, and Librarian. Administrators are the technical people who manage the sites; Staff work at our office and need some permissions on the back-end

but don't need to see the complexities of Drupal; and Librarians get some special privileges to add content and see various resources that we provide to them.

Assigning Permissions

In a larger library with a more complex organization, you want to take the time to fine-tune your roles and permissions. For example, you might want to provide a Web form that allows patrons to submit a reference question. You could have a role for staff working at the reference desk. You could then assign permissions so that only users who were given the role Reference Desk could read and respond to the reference questions. If you wanted to have departmental blogs on your site, you wouldn't want all your staff to be able to blog, just your designated bloggers. So you could create another role, Bloggers, and give that role permissions to blog on the site (see figure 11).

Permission	anonymous user	authenticated user	administrator	librarian
block module				
administer blocks	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
use PHP for block visibility	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
comment module				
access comments	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
administer comments	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
post comments	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
post comments without approval	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
filter module				
administer filters	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
menu module				
administer menu	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Figure 11
Drupal screen for assigning permissions.

In this scenario, you are going to have staff who work the reference desk and don't blog, staff who blog but don't work the reference desk, staff who do neither, and staff who do both. Fortunately, users can have multiple roles in Drupal, so one user can have the two roles Reference Desk and Blogger and get all the permissions assigned to those two roles, as well as any permissions granted to Authenticated users.

Managing security when roles overlap is slightly tricky; remember that Drupal only adds permissions to a role and does not block a permission for a role. Practically, this most often comes into play when you are trying to remove a permission from a user or a group of users on the site. For example, you might uncheck the permission Post Comments Without Approval for your Blogger role,

but if the Reference Desk role still has that permission, then any users with both roles will still be able to post comments without approval.

More concretely, permissions are set in Web UI. The Permissions page is organized as a long table of roles and permissions, divided up by which module is setting those permissions. This is not the easiest form to use, so don't despair; it may look overwhelming at first, but it will eventually make sense to you. Start by just reading through the permissions that are available and pragmatically decide who needs which permissions.

As a final note, any time you set up a site, you will want to create a test user with each role and test your site as these other users. Sites are often built as an Administrator user or even as User 1. This can give you a false sense that everything is working on your site. You also need to test each user and make sure that they are not getting permissions that were not intended.