**Lab 7: User-Contributed Content**

**20 points**

In this lab you will continue the configuration of Drupal, setting up roles, access controls, and permissions. In addition, you will install a WYSIWYG editor and set up a blog.

**Tasks**

1. Email me your preferred term project domain name (e.g. FirstCRC.org) and IP address. I'll set up the domain name e.g. FirstCRC.calvincs.com, and you can configure the appropriate server to respond to that domain name also for your dev site.
2. Before you start, back up your website databases by exporting their contents into files with commands such as

mysqldump -u root -p drupal\_*mysite* > *mysite-todaysdate*.sql

It's possible to mess up your web site, so you should make a backup before making significant changes. You can restore your website to an earlier version by deleting the database and restoring an older version. I won't add backup instructions to future labs—instead, you should make it a regular practice.

Similarly, before you upgrade a Drupal version, installing packages, etc. (or after you have made changes and your site is working the way you want it to), you should do a git commit. That is,  
cd /var/www/   
git add \*  
git commit

1. The remainder of this lab should be done on your lab website, though you'll do similar things on your dev site as well, as a part of your term project.

Install the Ubuntu package ftpd to enable you to use Drupal's update manager. In order for this to work, the FTP user you log in with must have write permission to the /var/www/html/sites/all directory. Now install the [module filter](http://drupal.org/project/module_filter) module using the web-based method, i.e. download the zip file into /var/www/sites/all/modules and unzip it. Enable module filter and update manager modules.

1. Now you should have a new link, "install new module", available on your module page. Use it to install install [ctools](https://drupal.org/project/ctools) and [sweaver](http://drupal.org/project/sweaver) for styling your website. While you are at it, install the CKEditor module [CKEditor](http://drupal.org/project/ckeditor)module (a WYSIWYG editor). You'll need to use an FTP login that has write permission to the /var/www/sites/all directory. Note: if you can't get this to work, you can always install modules the old way. This might be slightly more convenient.
2. Make sure you have Administrator, Editor, and Authenticated User roles set up on your server with appropriate permissions. In the next few days I will be creating three users on your lab web site: hpad, hped, and hpau. I'll also create hpau and hpad users on your dev website. When they have appeared, make the hpad users administrators, the hped user an editor, and leave the hpau users as a simple authenticated user.
3. Configure roles so that editors and authenticated users can access CKEditor and administrators can administer it.
4. Enable the blog module and create a sample blog entry. Make sure it is present on the /blog page.
5. Add a link to the blogs in your primary menu.
6. Set up permissions so that editors (and administrators) can create and edit their own blog entries.
7. Enable sweaver. Use it to customize some aspect of your theme. Then disable it again.
8. Install and enable Mollom or reCAPTCHA to control spam.

**Turn In**

I will check your web site directly from the link on the course page. I'll also check that you've given appropriate permissions to the users I create and that you've emailed me a domain name and IP address preference.

**Lab 8: Job Site**

**20 points**

In this lab you will get some practice with content types and Views by setting up a job board for your lab website. The job board will allow site users to submit and view job opportunities and to apply online. A custom views will show available jobs in selected departments.

**Tasks**

**Define a Job content type**

1. If you haven't already, enable Clean URLs for your website. This requires telling Apache2 to allow the use of .htaccess files (such as /var/www/html/.htaccess) to set configuration options. To do so, edit /etc/apache2/apache2.conf and in the section set AllowOverride to All. Restart apache and enable Clean URLs in the Configuration administration menu.
2. Install and enable [Entity,](http://drupal.org/project/entity) [Entity Reference](http://drupal.org/project/entityreference), and [Chaos Tools](http://drupal.org/project/ctools) [if it isn't already installed]. Also enable the Entity API module. Make sure Field, Field UI, Number, and List are enabled.
3. Create a new content type called "Job". Set the description to "A currently available job." Change the Title field label to "Job title". Disallow comments. Click "Save and add fields."
4. Now make some additions and changes to the fields.
   * Change the Body field label to "Description".
   * Add a field labeled Department of type List (text). Set the widget to Select List, Save, and make the allowable values
   * Administration
   * Business
   * Computer Science
   * Kinesiology
   * Physical Plant
   * Add an integer Salary field. Set the input widget to "Text field". Set the help text to "Enter an annual salary for this position.", the minimum to 0, and the field prefix to "$".
   * Add a Contact field. This will be an "Entity reference" field. Set the entry widget to Select list. On the next screen set the target type to User and click "Save..." Set the help text to "Select the person who is the primary contact for hiring for this position." Make it a required field.
   * Reorder the fields so that they these fields appear first in the list, in this order: Job title, Department, Description, Salary, Contact.
5. Click the Manage Display tab to adjust the way the fields will appear.
   * Make the Description field last and set its label to "Above". Make the other three fields "Inline".
   * Click the gear icon to the right of the salary field and set the thousands marker to a comma.
   * Click the gear icon to the right of the Contact field and check "Link label to the referenced entity".
6. Next, let's make some adjustments to the teaser view of a job. Still under Manage Display, click the "Teaser" button in the upper right.
   * Move the Department field to the top of the list and Salary just below it.
   * Set the salary thousand marker to comma again.
   * Set the labels to Inline for Department and Salary. Set the Description label to Hidden.
   * Set the Description trim length to 200.
7. Set user role permissions so editors will be able to create Job content, delete own job content, and edit own job content. Administrators should have all permissions for job content.
8. Add three example jobs from two different departments. In addition to title, salary, and contact person, put in job descriptions that are one or two sentences long.

**Define a Job Application content type**

1. Create a content type called Job Application with description "An application for a position." Set publishing options so that job applications are not promoted to the front page.
2. Change the name of the Body field to "Introductory Note."
3. Add a new field titled "Job" (machine name field\_jobref) of type Entity Reference. Use a select list as the entry widget. Set the target type to Node and click the Job checkbox.
4. Add a field titled "Resume" of type file. Enter help text "Upload your resume." Set the allowed file extensions to "pdf doc docx txt pages odf". Limit uploads to 2MB.
5. Order the fields Title, Job, Introductory Note, Resume.
6. Set up permissions so that administrators have all permissions on job applications and authenticated users can create them.

**Set up a job view**

1. Install the modules [Views](http://drupal.org/project/views) and [Advanced help](http://drupal.org/project/advanced_help). Enable Views, Views UI, and Advanced Help.
2. Next, we will create a view that allows us to browse the available jobs. Click Structure -> Views -> Add to create a new view titled "Jobs" of Content of type Job. Set the title to "Available positions", the path to jobs, and display format to Table. Also create a menu link titled "Available Positions" and an RSS feed. Click "Continue & Edit".

Next we'll adjust some settings on the main Views page. Note that you can scroll down to the bottom to see the preview at any point.

1. In the Fields section, the Title is already show. Add the fields Content: Contact, Content: Post date, and Content: Salary. For the Contact field, click "Link label to referenced entity." For the Content: Contact field accept the default display options. For the Post date field, set the Date format to "Time ago (with "ago" appended)." For the Salary field set the thousands marker to comma. Click the down arrow beside "add" and select "rearrange" to put the fields in the order Title, Post date, Salary, Contact.
2. Note that we are currently displaying jobs in all departments. We want to add a contextual filter to allow us to select the department. Click Advanced to show the advanced options. Add a contextual filter, and
   * Select Content: Department.
   * Click to Display a Summary when the filter value is not in the URL
   * Click the "Override title" checkbox and enter "Jobs in the %1 department".
   * In the More section, set Case to Capitalize each word, Case in Path to Lower case, and click Transform Spaces to dashes in URL.
3. Click to save your view. In the preview section, enter a department name in the box and update the preview. Note the change in the jobs listed. Also check out the *yourwebsite*/jobs page.

**Set up an applications view**

1. Add a new view with settings as follows:

|  |  |
| --- | --- |
| **View setting** | **Value** |
| View name | Applications |
| Show | Content of type Job application |
| **Page Settings** |  |
| Display format | Table |
| Create a menu link | Checked; Menu: Navigation menu |

1. Add the fields Content: Job and Content: Post date. Select the date short format.
2. We also want to add the name of the applicant, but it's not in the list. There is a field Content: Author uid, but the message under the field says "If you need more fields than the uid add the content: author relationship." Let's do that.

In the advanced section add a relationship. Select Content: Author and click Apply. Now you can add the User: Name field to the list of fields. Put the fields in the order Post date, Title, Name, Job.

1. Job applications are obviously sensitive, so set it so that only editors view applications. In the page settings section, click Permission. Click Role, then Editor. Save your changes.
2. Click Save to save your view.
3. Add a job application with a random file attached as a "resume". Follow the link in the navigation menu to view it.

**Turn In**

I will check your lab website directly from the link on the course page.

# Lab 9: Event Management and Calendar

**20 points**

Suppose you were preparing a website to keep track of a dorm calendar of events. Members might want to be able to see when events are happening on a calendar and post their own events to the site. To make it easy to see what is coming next, there should be a short block of upcoming events that could be placed on a homepage. The calendar should offer daily, weekly, and monthly views as well as a way to subscribe to the calendar using Microsoft Outlook or Apple iCal. Finally, it would be good if there were a way to track those who are planning to attend.

In Lab 9 you will set up this event management system on your lab server.

**Tasks**

1. For this lab you will need to install the modules [Date](http://drupal.org/project/date), which provides a date field for content types, [Calendar](http://drupal.org/project/calendar), which can display events in a calendar format, and [Date iCal](http://drupal.org/project/date_ical), which can set up an iCal feed. You'll also need [Libraries](https://drupal.org/project/libraries). Make sure the following modules are enabled: Libraries, Date API, Date, Date All Day, Date Popup, Date Views.

**Event content type**

1. Create new new content type called Event, with description Dorm event, and fill in appropriate values:
   * Title field label: Name
   * Don't promote to front page by default
   * Under comment settings, uncheck "show reply form on the same page as comments"
   * Body field label: Description
   * Add a text field called Location for storing the location of the event.
   * Add a field called Time of type Date with the pop-up calendar entry widget. In the field settings, check "Collect an end date" but don't make it required. On the next form, make Time a required field. Under More Settings and Values, select appropriate date entry options. Change Starting Year to -1 year from now.
   * Put the fields in this order: Name, Time, Location, Description
2. Set up access control appropriately. Give authenticated users and higher the ability to add events.
3. Create three example events on different days, two upcoming (one at least a month in the future) and one more than a day in the past.

**Event view**

1. Add a new node view named "Upcoming Events" with the description "A block of upcoming events". It should show content of type event. Set it to create a block, not a page, with display format HTML List of titles (linked). Click Continue & Edit.
2. In the Fields section, add Content: Time, but don't display the label "Time:" Display the time in the Medium format.
3. Now we want to configure the view to show upcoming events: Add a filter on the **Content Time - start date** field and Click apply twice. Set the operator to "is greater than or equal to", set the date selector to "Enter a relative date," and type in "now". You should see only future events in the preview.
4. In the Sort critera box, add a sort on **Content: Time** (ascending).
5. Save this block and add it to a sidebar on the site's home page.

**Calendar view**

The calendar view module provides a rich new View type called Calendar, but it requires various arguments to be set up properly. For example, if the calender URL is *calendar*, the Calendar view will handle URLs like calendar, /calendar/month, etc.

1. Enable the modules Calendar and Date iCal.
2. On the Views administration page, click "Add view from template". Click "add" beside "A calendar view of the 'field\_time' field in the 'node' base table." Click continue.
3. On the View configuration page, make sure the Month display is selected, and change the title to "Calendar of Events." Under Page Settings, change the Path: to /calendar/month". Save your settings.
4. Test the URLs /calendar/month and /calendar/year.
5. Use Structure -> Menus to add a menu item to the main menu labeled Calendar that opens the monthly calendar view.

**Taking it further (optional)**

Do you want to track who will attend events? Let's add an "I will attend this event" link at the bottom of each event, together with an "Attendees" view for each event.

1. Install the [Flag](http://www.drupal.org/project/flag), module, which allows you to add on/off switches to items. Enable it.
2. Go to Structure -> Flags and edit the existing bookmark. Change its name to attendance and its link text to "attend this event." Change its unlink text to "cancel attendance". Fill out other options appropriately.
3. Create a view that will list the attendees for any event. You'll need to add a relationship to "Flags: User's flagged content" to get access to the flag information. You'll also need to add an argument, Flags: Content ID.

**Turn In**

I will check your website directly from the link on the course page.

# Lab 10: Creating a Custom Theme

**Summary**

You can customize the look of your website by creating your own theme. The easiest way to do this is to start with an existing theme that is similar to what you want and changing it as needed to get the exact look you are seeking.

In Lab 10 you will create your own theme which will be a customized version of an existing theme.

**Tasks**

1. For this lab you will customize the Bartik default theme, which should already be installed. First, copy the entire /var/www/html/themes/bartik directory directory into sites/all/themes and change the name of that directory to "newbartik".
2. The .info file holds basic information about the theme such as its name. Change the name of the .info file in the newbartik directory from bartik.info to newbartik.info and open it in a text editor. Change the name line to reflect the new name. Change the description appropriately. Enable the NewBartik theme and make it the default for your lab site. (Make sure the Upcoming Events block is still visible on the home page.)
3. The template.php file contains some php functions for the theme to process pages, handle variables, etc. Edit this file and replace all instances of "bartik" with "newbartik".

**Changing the CSS**

1. Drupal will optionally combine all CSS files together into one big file for performance reasons. However, this will prevent you from seeing changes you make to CSS files. For this lab, click on Configuration -> Performance and make sure the "Aggregate" options are disabled. Turn off caching.
2. Let's put a box around content items on the home page to make them stand out. Open NewBartik's style.css file in a text editor. Locate this rule (line 584):  
   .node-teaser .content {  
   font-size: 1em;   
   }  
   and add two new rules:  
   border: 1px dashed #ccc;  
   padding: .5em;  
   Save the file and reload the page.

**Modifying template files**

Modifying CSS can get you a long way, but for some changes you will need to modify the HTML generated by your theme. Nodes, comments, and other content types are generated by template files, which you can customize. For example, comments are generated by *comment.tpl.php* and the entire page is generated by *page.tpl.php*. These files are mostly HTML with a little bits of PHP to print out the contents of elements like titles, breadcrumbs, etc. We'll modify the page template to move breadcrumbs into the page header area.

It can be difficult to figure out what templates generate each section of a page. Fortunately, there is a way to configure Drupal to give you some help along those lines by adding a setting in the settings.php file. Add the following line at the end:

$conf['theme\_debug'] = TRUE;

Settings.php file in sites/default/settings.phps

Now restart your server and view the source of your home page. In source of your home page, search for "logo". Now scroll back a couple of lines. You should see in the comments that theme debug mode adds that the template that generates this section is "page.tpl.php".

1. Go to the home page and then to another page, e.g. "Available Positions." Notice the breadcrumb ("Home") near the top of the content area. Let's move it into the header area, below the slogan.
2. Edit the file page.tpl.php in the newbartik/templates directory. Go to line 117 and see where the name and slogan are being displayed. Now go to line 177 and notice where the breadcrumb is being printed. Move the line printing the breadcrumb so it appears inside the header section on line 120, just before the </div>. Reload the page and observe the difference.

**Theming specific content types**

It is possible to customize the way specific content types are displayed by edting the appropriate template file. For example, to customize the way comments are displayed, you would edit comment.tpl.php. Let's modify the way articles are displayed to display author submission information below the story. Since there is no article.tpl.php file already, we'll create our own by copying node.tpl.php.

1. Create a new article with a title and contents. Note the position of the submission information.
2. Copy node.tpl.php to node--article.tpl.php and open it in a text editor. Note that the user picture and submission information appear before the contents of the story. Move them after the contents of the story. View the story again and note the new position of the submission information. If you don't see any changes, go to Configuration > Performance and clear all caches. Then go back and reload the page.

**Adding or overriding template variables**

Drupal generates many useful variables for you, like $site\_name and $submitted, but what if they don't generate the exact output you want? A process\_page hook lets you run code when a page is generated to modify available variables or creates new ones (The Drupal site offers much more information about these [advanced overrides](http://drupal.org/node/173880).)

1. We will add a line to the newbartik\_process\_page() that may be found in the template.php file to add a new variable. Edit that file and find the appropriate hook (function). Add the line   
   $variables['random\_number'] = rand(1,100);  
   at the beginning of the function. Now put the random number in the header section of each page by putting <?php print $random\_number; ?> in the page template. Don't forget to clear the cache to see the change.

**Optional challenge**

1. Change the "Submitted by" message to "Offered by" for job postings.

The Drupal.org website has much more information about customizing templates. One place to start is the [customization snippets](http://drupal.org/handbook/customization/snippets) page, which has small cookbook recipies for customizing themes and modules.

**Turn In**

I will check your lab website for the NewBartik theme with the changes we made in this lab.

# Lab 11: Deployment and Revision Control

**20 points**

In this lab you will deploy your **lab** website on dreamhost. If the domain name for your lab website is abc123.calvincs.com, I've set up a subdomain deploy.abc123.calvincs.com that points to a dreamhost server. You'll deploy a copy of your lab website on the dreamhost server.

You will also experiment with the popular **git** source control system.

(I'll also set up hosted dreamhost subdomains as deploy.[yourproject].calvincs.com and link them on the projects page. You should deploy your term project websites to those accounts separately, not as a part of this lab.)

**Deploying your website**

Deploying a Drupal website on a hosting service generally includes three steps:

* Create a new directory deploy.abc123.calvincs.com in /var/www/html/sites and edit the new settings.php file with username, password, and server name appropriate for logging into the host's database server
* Copy your /var/www/html files into the host's server web directory (deploy.abc123.calvincs.com)
* Copy your database onto the host's database server

For this lab you'll deploy your lab website on the deploy.abc123.calvincs.com subdomain. I've also created a database for you on dreamhost with the login and password given to you. The database server is mysql.calvincs.com and the database name is *yourID*, that is, your login, e.g. zzz99. Let's start by creating a new drupal site to be uploaded to dreamhost.

1. Create a new drupal site directory /var/www/html/sites/deploy.abc123.calvincs.com by copying your default site.
2. cd into the new site directory you just created and edit the settings.php file to connect to your database on the dreamhost database server, mysql.calvincs.com (Note that localhost won't work as the host--you'll have to change it to mysql.calvincs.com.) Also change the database name, user, and password to those given to you.
3. The web server on dreamhost.com isn't configured to default to index.php files, so let's copy the index.php file to index.html in your webserver home directory with the command   
   cp /var/www/html/index.php /var/www/html/index.html
4. A single zipfile uploads much faster than thousands of separate files. Create a zip archive of all the files:   
   cd /var/www/html  
   zip -r /tmp/site.zip .htaccess \*  
   This creates the file "site.zip" in /tmp with all the contents of your /var/www/html directory. (The hidden .htaccess file has to be named separately because \* doesn't include it.)
5. Upload it to your dreamhost server with the command  
   scp /tmp/site.zip mylogin@william-blount.dreamhost.com:  
   While you are waiting for the upload, you can continue with the rest of the lab.
6. Log in to the william-blount.dreamhost.com server with ssh, cd into the website directory (deploy.abc123.calvincs.com), and unzip your files with the unzip command in the current directory. (You should end up with several files in deploy.abc123.calvincs.com including authorize.php, index.php, includes, sites, etc.)

**Copying your database**

1. Use mysqldump to create an SQL dump of your database. As a reminder, you can find instructions on the Resources page (mysqldump command). Copy copy the file to william-blount with the scp command, log into william-blount, and type the command  
   mysql -u drupal7user8 -p -h mysql.calvincs.com *[database name]* <*[mysqlDumpFile.sql]*
2. Set a cron job for Drupal as in [lab 6](http://cs.calvin.edu/curriculum/is/337/hplantin/labs/lab6.html).
3. Your website should be deployed and running. Give it a try!

**Revision Control**

We're simply going to use git locally, on your VM. You can also use git to synchronize your directory of files with other repositories or a server. If you eventually work at a company that uses source control, they will already have a server set up. If you want to share your code with others or collaborate on a project and don't have access to a server, you can set up an account at github.com -- for free, if the project is open source.

Note that there are git GUIs which may be easier for some purposes. A google search should locate several. However, it's a good idea to start out with the command-line interface in order to better learn how it works.

1. Type git help to see a list of common commands.
2. Introduce yourself to git. Type the following at a command prompt:  
   git config --global user.name "Your name here"  
   git config --global user.email yourID@students.calvin.edu
3. Your /var/www/html directory should already be under source control with git. Change into /var/www/html and type 'git status'. It should show you any modified and untracked files that are not current in your last git commit. If for some reason you don't have the directory under source control you can set it up now: type git init. Git should reply Initialized empty Git repository in .git/. Now add the current directory to the repository with the command git add . (the dot is part of the command). The snapshot is now stored in a temporary staging area which git calls the "index".
4. To upload the contents of your www directory to the git server type git commit -m "Lab 11 commit". The -m option gives the *commit message*explaining the changes you are committing. Congratulations! You've backed up your code and made it available to everyone else in your group (if there were anyone else in your group).
5. Edit the file robots.txt in the webserver root directory (/var/www/html). Robots (Google's Web crawler, for example) read this file to learn what they may crawl and what they may not. Since this is a development site, we don't want search engines to index it. So, after the User-agent: line add a line that says   
   Disallow: /  
   This will prevent (obedient) robots from crawling the site.
6. Type git commit -a. (The -a option means you don't have to manually add the changed file with a git add command.) Add a suitable commit message, such as "Prevent robots from indexing the site".
7. Type git log to see recently changed files. Now try git log -p to see all the actual changes.
8. Let's create a new branch of the repository. Type git branch experimental. If you now type git branch you'll get a list of existing branches. The master branch is still active.
9. Let's switch to the experimental branch. Type git checkout experimental. Now edit the robots.txt file and add a comment at the top that says   
   # this is the experimental branch  
   Commit the change.
10. Switch back and forth between the master and experimental branch and watch the contents of that file change.
11. Now switch back to the master branch with git checkout master. Edit the robots.txt file and check that the change you made is no longer visible since it was on a different branch. Now edit the robots.txt file and make a different change -- add the comment   
    # this is the master branch  
    Commit your change.
12. At this point you have two different branches. Let's merge them back into one. Type the command  
    git merge experimental  
    You'll get a message about a merge conflict -- there are two different versions of the robots.txt file.   
    git diff will show you the differences. Edit robots.txt to remove the stuff that was added, including the comments, and type git commit -ato commit the result of the merge.
13. Edit your robots.txt file and make sure the Disallow: / line is deleted. Commit your changes.
14. This lab was intended to introduce some revision control concepts. It was based in part on the [git tutorial](http://www.kernel.org/pub/software/scm/git/docs/gittutorial.html) at kernel.org. There is more to learn about git if you intend to use it. You can optionally continue with that tutorial or another to learn more. If you wish to upload your website to dreamhost using git, see their git [wiki page](http://wiki.dreamhost.com/Git).

**Turn In**

I will check your git repository and the deployment of your live site on*yoursite*.calvincs.com