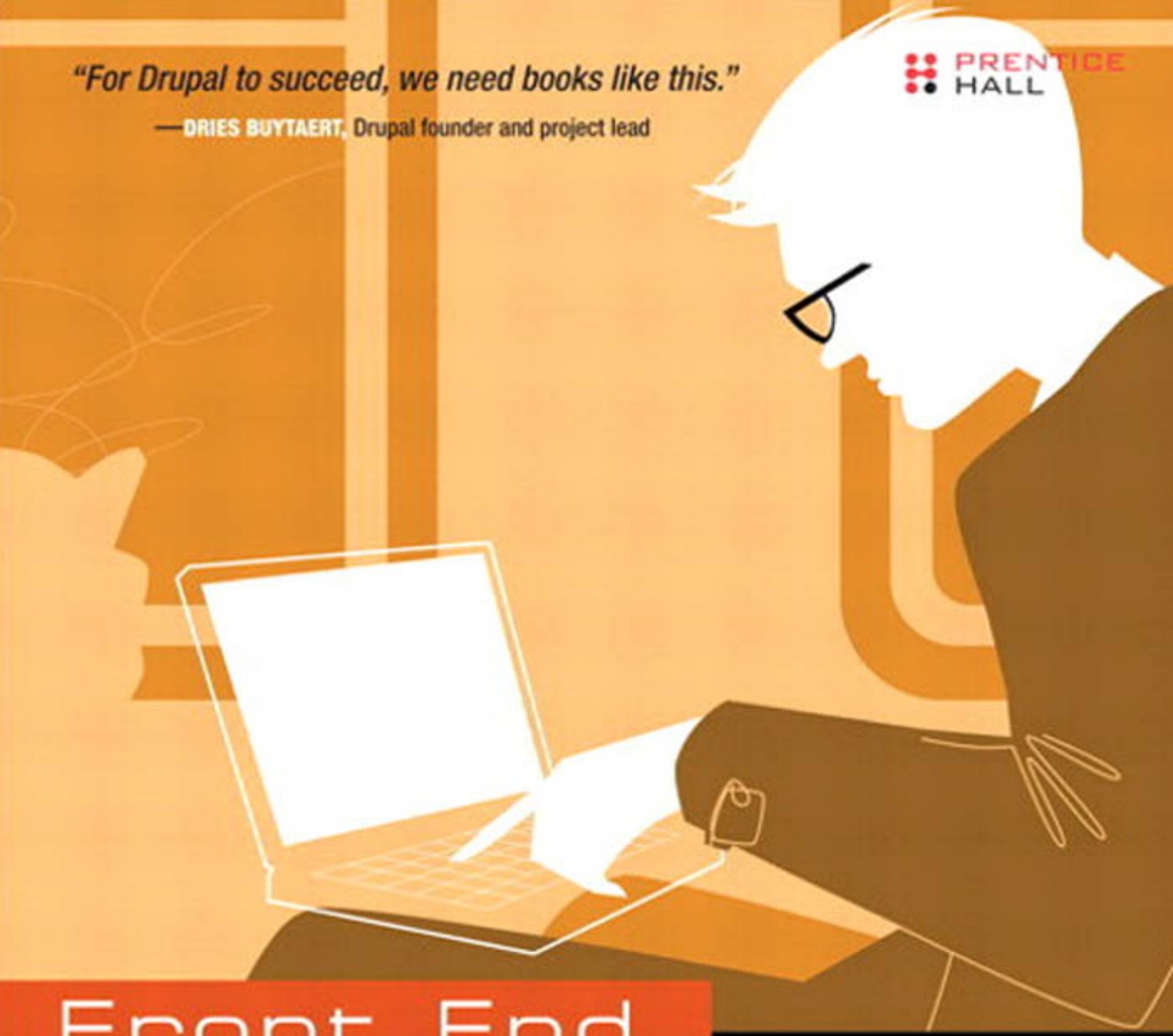
"For Drupal to succeed, we need books like this."

—DRIES BUYTAERT, Drupal founder and project lead

PRENTICE HALL



Front End Drupal

DESIGNING, THEMING, SCRIPTING

EMMA JANE HOGBIN
KONSTANTIN KÄFER

Library of Congress Cataloging-in-Publication Data

Hogbin, Emma Jane.

Front end Drupal : designing, theming, scripting / Emma Jane Hogbin and Konstantin Käfer.

p. cm.

Includes index.

ISBN 978-0-13-713669-8 (pbk. : alk. paper) 1. Drupal (Computer file) 2. Web sites-Design-Computer programs. 3. Web site development. I. Käfer, Konstantin. II. Title.

TK5105.8885.D78H65 2009

006.7'6—dc22

2009002636

ISBN-13: 978-0-13-713669-8
ISBN-10: 0-13-713669-2
Text printed in the United States on recycled paper at R.R. Donnelley in Crawfordsville, IN.
Second printing, June 2009

Editor-in-Chief
Mark Taub

Executive Editor
Debra Williams Cauley

Development Editor
Songlin Qiu

Managing Editor
John Fuller

Project Editor
Anna Popick

Copy Editor
Jill Hobbs

Indexer
Michael Loo

Proofreader
Linda Begley

Technical Reviewers
Károly Négyesi
Bernie Monette
Lynda Chiotti
Caroline Hill
R.G. Daniel

Cover Designer
Chuti Prasertsith

Composition
Gloria Schurick

Graphics
Tammy Graham
Laura Robbins

Foreword

At DrupalCon Barcelona in 2007, while giving my regular "State of Drupal" presentation, I remarked that during my hour-long session, four new Drupal sites would be launched. I went on to suggest that three of those four sites would be ugly. A year later, at DrupalCon Szeged in Hungary, those four new sites per hour had grown to seven and Drupal 6 had been released, making it easier to create great-looking Web sites. Still, even now, Drupal faces a common problem on the Web—the relative lack of new, high quality themes.

Front End Drupal tackles that problem directly and is designed to help both experienced designers and rank novices get an understanding of how Drupal theming works. From using contributed "starter themes," to customizing templates to modify the markup used in Drupal's output, to using jQuery and JavaScript to enhance the user experience, *Front End Drupal* clearly charts a path to theming mastery. In fact, I'll be the first to admit that I learned a lot from this book.

The Drupal community has created a remarkable platform that powers sites of all sizes and descriptions, all around the world. Together, we've crafted a robust, extensible content-management system that illustrates some of the key values in our community: flexibility and utility, innovation and openness. But Drupal has always been a developer's platform, even with the many designers in our ranks. It's about time those designers had a great book. In fact, this book is valuable not just to the designers we have, but to the designers we want—the thousands who have never worked with Drupal.

The thing is that creating a Drupal theme isn't always easy. It's a crosscutting experience that requires a lot of diverse skills and utilizes expertise in XHTML, CSS, JavaScript, and PHP, all within the context of Drupal. Doing a Drupal theme right can be challenging, but it is also exciting and incredibly rewarding. A survey I conducted in 2008 listed "Finding skilled Drupal designers" as the number one entry on

the list of the "Top five most difficult things," as reported by both expert and novice users. We need to do more to find new themers, as well as encourage and support the ones we already have.

I'm excited that Emma Jane and Konstantin recognized that and authored this book. It fills an important need in the Drupal ecosystem and will bring a new attention to design in Drupal. Since I've mostly focused on the "back end," it's nice to see the "front end" get more and more attention. For Drupal to succeed, we need books like this. We need the skills it teaches and we need the people it attracts. We need the new themes those people will create and the new suggestions and improvements they bring to our project.

Dries Buytaert
Drupal founder and project lead

Preface

Drupal is an open-source content management system software package that is free to download, modify, and use. It has been implemented by thousands of people around the world and is used by millions of people daily as the basis for discussion Web sites, community portals, corporate intranets, e-commerce Web sites, vanity Web sites, resource directories, image galleries, podcasts, and more! By choosing to use Drupal, you are accessing not only an award-winning Web platform, but also its vibrant community.

This book will teach you how to customize how Drupal looks. Applying new designs is very easy—the code that controls how Drupal *works* is separated from the code that controls how Drupal *looks.* The design part of Drupal is referred to as the theme layer—and that's what this book is all about. Individual designs are referred to as "themes" and the people who create and implement them are referred to as "themers." By the time you reach the end of this book, you will have the tools to customize the experience for your content managers, Web site visitors, and Drupal administrators.

The book assumes you are familiar with how Drupal works and that you have been an administrator of a Drupal Web site. It would help if you are comfortable with Web site design and development, but these concepts will be explained for those who have only a limited experience with them. More specifically, this book will use code snippets written in HTML, CSS, PHP, and JavaScript.

Chapter 1

This chapter covers the basics of Web page design. It will help you to prepare your information so that it will slide easily into a Drupal Web site. You will learn how to describe content and its organization; structure page layouts so that all of your interface components fit sanely onto your Web pages; and implement a work flow that works for your Drupal team.

Chapter 2

With the basics of Web design under your belt, it is time to prepare your workstation for Drupal theming. In this chapter, you will learn about Drupal terminology and theming strategies as well as must-have modules and browser tools. Chapter 2 also includes language references for each of the machine languages used in creating a Drupal theme.

Chapter 3

You will now move on to learning the basic anatomy of a Drupal theme. In Chapter 3, you will learn how to find and install a premade Drupal theme. You will also learn the anatomy of a Drupal theme and discover how to use Starter Themes to reduce your development time. Tips are included on how to convert themes from WordPress, Joomla!, and Drupal 5.x.

Chapter 4

The overall structure of pages in Drupal is defined by the page template. In this chapter, you will learn how to customize every part of this template—from using sitewide page variables and menus, to changing page templates based on the section you are currently in. Information on print-friendly templates and mobile devices is also included in this chapter.

Chapter 5

It's time to get to the guts of your Web site—so in Chapter 5, you will learn how to customize your Web site content, including individual nodes and teaser summaries. This chapter also describes the most appropriate image module to use for your Web site. Examples of output are provided to help you make the best decision for your content.

Chapter 6

The most commonly overlooked area in Drupal theme design is content editing forms. In this chapter, you will learn simple tips and tricks to make your forms more usable and will get a gentle introduction to altering forms with the Form API. Techniques described in this chapter will help you to enhance the usability of your content editing forms.

Chapter 7

If you are running a community site, this chapter is a must—it includes information on how to theme user profiles, community comments, and user-generated content. Additional information is provided on creating private, member-only sections to your Web site.

Chapter 8

In this chapter, which covers administrative interfaces, you will learn how to make the administration of Drupal a little bit easier. Techniques include creating custom administrative interfaces, adding task-based navigation, creating administrative menus, and customizing your Web site's error messages.

Chapter 9

In this chapter, you will acquire the JavaScript skills required for writing truly stunning, portable, and flexible components for your theme. Basic concepts or advanced object orientation—there's certainly something you'll learn in this chapter.

Chapter 10

An introduction to jQuery, the JavaScript library that ships with Drupal, will bring you up to speed with today's most prevalent JavaScript library. You'll also learn how jQuery is used in Drupal, how you can create stunning animations, and how you can implement AJAX callbacks to the server.

Chapter 11

In this chapter, you will learn how to apply your newfound JavaScript and jQuery knowledge to a Drupal Web site. By creating a horizontal scroller component, you'll learn step by step how to architect a highly flexible and reusable JavaScript widget. Additional information in this chapter includes server-side JavaScript integration and an excursion into the vast supply of ready-made jQuery plugins.

Appendices

Information on how to install Drupal and contributed modules is included in Appendix A. Appendix B contains the code samples that are referenced in the JavaScript chapters. These code samples can also be downloaded from the book's Web site.

Chapter 4

The Drupal Page

Get out your crayons and your coloring book! In this chapter you will learn how to connect the dots and build context-sensitive page templates. The adventures in this chapter begin by dissecting how Drupal builds the pages that are delivered to your Web browser. You will then learn about sitewide variables so you can split your page templates into a clean HTML framework with Drupal-served data being injected into the right spots at the right times. Next, you will learn to draw "outside the lines" with custom page variables and page templates based on categories, page aliases, and content types. And for those who don't like to color at all, the chapter wraps up with information on creating print-friendly templates and building a mobile-friendly clone of your Web site. In this chapter you dive into the guts of a Drupal theme. Note that the code snippets included here require a basic understanding of PHP, CSS, and XHTML.

Elements of a Page

When you understand how Drupal builds its themes, it becomes very easy to achieve complicated tasks. A common question is, "I need to inject a block into the content of the front page—how do I do that?" This is not how Drupal thinks about

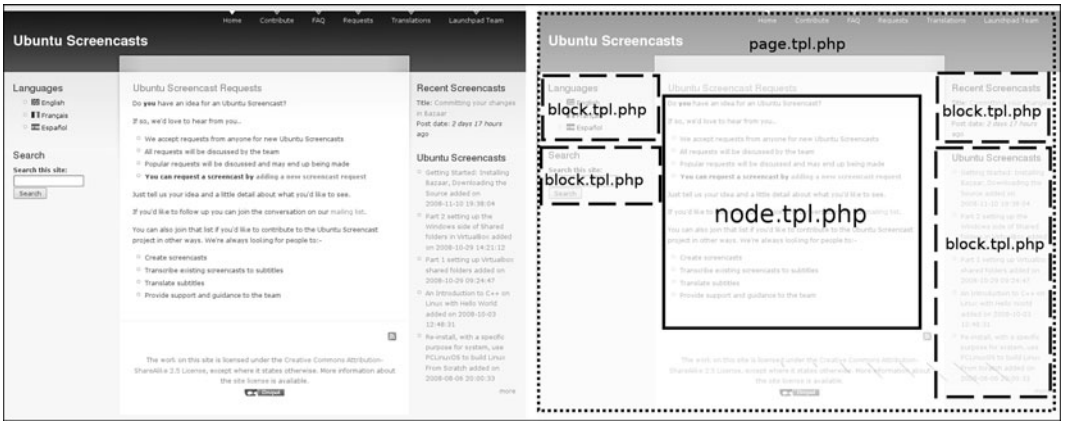


FIGURE 4.1 The Drupal page is customized by using many different templates.

this problem, so the answer seems very difficult. Instead of thinking about the page as it appears in the Web browser, you must think about each of the elements separately. Figure 4.1 illustrates how Drupal customizes a page with each of its template files.

The whole page is controlled by the template `page.tpl.php`. Within the whole page, several more template files are injected to customize each of the different components. These templates theme the output from various modules within Drupal. Block and node templates are shown in Figure 4.1. Each module that outputs content to the page will have its own templates, which you can in turn customize.

Dissecting a Theme

Most themes include a customization of the page, block, and node templates, which are the main building blocks that are used to construct the layout of a page. If you are working with a downloaded theme, look in your theme's directory for the following files:

- `page.tpl.php`
- `block.tpl.php`
- `node.tpl.php`

These three files are the building blocks that define the markup of your site. In-depth information on customizing `page.tpl.php` appears later in this chapter, and additional information on customizing `node.tpl.php` can be found in Chapter 5.

Here is another analogy for thinking about the Drupal page template: It is a little bit like a large parking garage with numbered spaces. The garage itself does not care which kind of car or truck or motorcycle is parked in each space; it merely houses the lines that show each of the areas where a vehicle might fit. The garage might have different colors for each of the levels to make it easier for people to remember which level they are parked on. The people who operate the garage may have rules about which space each person may park his or her vehicle. It is impossible to park your vehicle in two places at the same time in the parking garage.

In Drupal terminology, the page template defines regions (levels in the parking garage) where blocks may appear (assigned spaces for parked vehicles). A single block may not appear more than once in a page (cars may be parked in only one space at a time); however, this region can change location within the page template depending on the context (parking garages may have different colors for each level in the garage). Later in this chapter you will learn how to assign new blueprints to your "parking garage."

This analogy is not a perfect one, of course: In real life, a vehicle can park somewhere other than its assigned place. In contrast, blocks in Drupal may be assigned only one spot throughout the Web site. Nevertheless, the parking garage analogy is a helpful way to think about how the page template keeps order without being aware of the displayed content of a page.

In Chapter 3, you created with a basic page template that contained only Drupal output and a skeleton HTML framework. You will now start to build on these basics to create a more sophisticated page template.

Sitewide Page Variables

The variables available in the template file `page.tpl.php` are classified into several categories:

- **General utility variables** are used to build context-sensitive templates with directory names relevant to the path of the theme's location on the server.
- **Page metadata** includes page language, style and script tags relevant to the page, and body classes.
- **Site identity** takes the form of the site name, site slogan, site mission, and logo.

- **Navigation** includes items related to primary and secondary navigation, as well as search boxes.
- **Page content** includes the page title, dynamic help text and Drupal system messages, and tabs.
- **Footer and closing data** includes RSS feed icons, footer messages, and final markup from any modules ("closure").

Commonly used variables are identified in Figure 4.2, which depicts a fresh installation of Drupal, using the theme Garland.

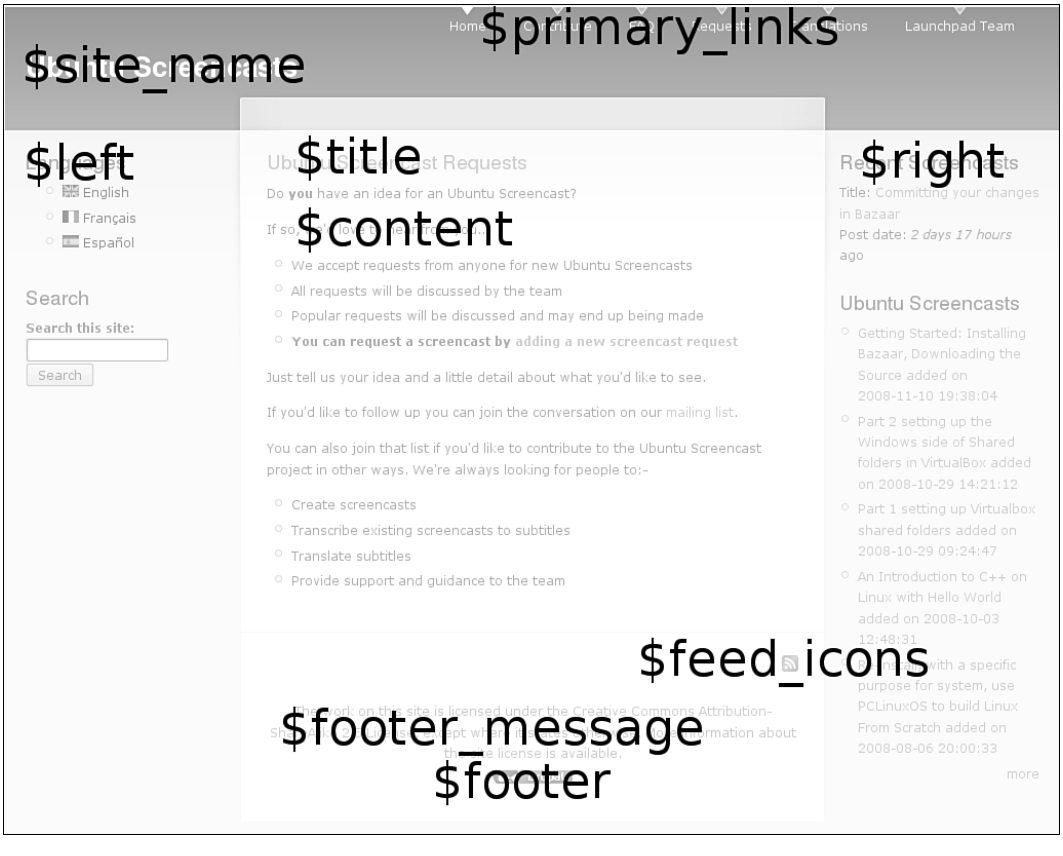


FIGURE 4.2 Common variables displayed in the Garland theme.

A complete list of page template variables is available from the Drupal directory `modules/system`, in the file `page.tpl.php`, which is also available online at `http://api.drupal.org/api/file/modules/system/page.tpl.php`.

General Utility Variables

The general utility variables represent a very basic toolkit with which you can customize your site's template based on the characteristics of the visitor. They include the following variables:

- Variables useful in linking to images and files within your site, such as `$base_path` (the base URL for the Drupal installation) and `$directory` (the base directory for this theme)
- `$is_front`, which reports if the current page is the front page of the site
- User status checks, including the test of whether the visitor is logged into the site (`$logged_in`) and whether the user has access to administration pages (`$is_admin`)

Page Metadata

The page metadata variables are used in the `<head>` tag of the page template. This set includes the following variables:

- An object containing the language the site is being displayed in. To print the text representation of the language to your template, use the following variable: `$language->language`.
- `$head_title`: A modified version of the page title containing the site name, for use in the `<title>` tag.
- `$head`: Metadata for metatags, keyword tags to be inserted into the `<head>` section.
- `$styles`: Style tags used to link all CSS files for the page.
- `$scripts`: Script tags used to load the JavaScript files and settings for the page.

In addition to this metadata, there is a wonderful variable that contains a set of conditions to help you style each page: `$body_classes`. The `$body_classes` variable includes the following information: the current layout (multiple columns, single column); whether the current visitor is an authenticated user; and the type of the

node being displayed (for example, `node-type-book`). This variable includes only the names of the classes to be used by your style sheets. To use it in your theme, you must include the following PHP snippet:

```
<body class="<?php print $body_classes ?>">
```

Site Identity

The site identity information comprises a set of variables that outputs information about your site. You can alter the contents of and/or disable each of these variables in Drupal's administration area by navigating to Administer, Site configuration, Site information.

- `$front_page`: The URL of the front page. Use this variable instead of `$base_path` when linking to the front page. It includes the language domain or prefix.
- `$logo`: The path to the logo image, as defined in the theme.
- `$site_name`: The name of your Web site.

Two other variables can be set within the site identity section of the Drupal administration area:

- `$site_slogan`: The slogan of the site.
- `$mission`: The text of the site mission.

There is no rule that says you must use these last two variables for their intended purpose; in fact, you can use them to store any information you would like to display within your page template.

Page Content, Drupal Messages, and Help Text

Content is the most important part of your Web site. You must tell Drupal where to insert content into the page template! This is done with a simple variable, `$content`. You may place this variable anywhere in the template file `page.tpl.php`. From this simple variable, Drupal may present a single node, or a list of nodes, or whatever else Drupal may prepare as the "content" for any given page.

You must also print the title for this content using the variable `$title`. It is different than the variable `$head_title`, which includes the name of the Web site and is typically printed in the `<title>` tag for a page.

There are two modes for each node: view and edit. These modes can be accessed through the tabs that are displayed on each node. Within your page template, the variables `$tabs` (primary level of tabs) and `$tabs2` (subnavigation available present in several administrative pages) are used to place links that access the "view" and "edit" modes for each node. The tab variables are typically printed between the `$title` and `$content` variables.

Breadcrumbs
Although there is a variable containing the breadcrumb path for each page, the breadcrumb trail is often incomplete. Many themes choose to display this variable only in the administrative section of the Web page.

Drupal communicates system messages to the user through the variable `$messages`. This variable may contain useful information that describes the successful submission of new content or content modifications, errors relating to a form submission, or messages within the administration system. Messages come in three flavors: *status, warning,* and *error.* Through your style sheet you can make these messages visually unique. Typical colors used for these messages are green for status messages, yellow for warning messages, and red for error messages. The messages are available as CSS classes and carry the corresponding name (for example, warning messages use the CSS selector `.warning`).

In addition to these system messages, Drupal will occasionally provide "help" text, which is made available through the variable `$help`. Both the help text and messages must be specified in your page template to ensure that the appropriate system messages are delivered to your Web site users.

Creating New Page Variables

In addition to using the variables that are provided by Drupal, you can create your own. Each time Drupal builds a page, it gathers the information it needs to display that page and makes sure the information is safe to display. This "preprocessing" is completed before the page is built using the template files. To keep your template files focused only on HTML output, you can insert any custom programming you need into the relevant preprocess function. Its output will be returned as a variable to the relevant `tpl.php` template file. Variables created in the preprocess functions are available only in the relevant template files (`tpl.php`).

Placing PHP snippets into templates

Throughout the rest of this chapter, you will be working with preprocess functions and creating new theme variables. The preprocess functions are always placed in your theme's `template.php` file. Theme variables are always placed in the relevant template file (for example, `page.tpl.php`).

Preprocess functions are named according to the template you want to "hook" your new variables to. Any module that has a template file can use the preprocess function. For example, the page, node, comment, and block types all have associated `.tpl.php` files; as a consequence, they can all be tied to a preprocess function. A full list of preprocess functions is available from the API documentation at `http://api.drupal.org/api/search/6/preprocess`. More information on creating additional template files is provided later in this chapter.

In the following example, you will add a new variable that can be used in the template `page.tpl.php`. Your imagination is the only limit on what these variables can contain! The Zen theme inserts additional, sophisticated body classes that allow you to create very specialized page customizations through CSS. The Garland theme uses a preprocess page function to hook into the color module. Later in this chapter, you will learn how to add new image banners based on which section of the Web site you are viewing.

In this example, we will add a new graphic to the page if the visitor is logged into the site but is not currently viewing the front page.

```
function bolg_preprocess_page (&$variables) {
   // Add a "go home" button to page.tpl.php

   if ($variables['logged_in'] == TRUE && $variables['is_front'] == FALSE) {
      $image_path = $variables['directory'] . "/images/go_home.jpg";
      $image_text = t("Go home!");
      $image = theme('image', $image_path, $image_text, $image_text);
      $variables['go_home'] = l($image, "<front>", array('html'=> TRUE));
   }
} // End of the preprocess_page function
```

In the file `page.tpl.php`, you can now place the new variable `$go_home` anywhere you would like the button to appear. Although the snippet could be simplified by

hard-coding the HTML for the image, this method can be easily reused in many different themes and allows the text string to be translated for multilingual Web sites.

Modifying Page Variables

You may also choose to modify variables that have already been set by Drupal. The Zen theme uses this technique to remove the markup for an empty help message. The Newswire theme customizes page variables to modify the HTML for the content title depending on which page is being viewed; Newswire also customizes the logo that is displayed on the front page and the inner pages of the site. The Acquia Marina theme removes the markup for sidebars when they are not in use to create a clean, collapsible template layout. You can implement your own customizations as well.

To reset a variable, simply use the same variable name as an existing page variable. Do not unset unused variables, as this action may cause an ugly PHP error if the `page.tpl.php` file tries to print a variable that no longer exists. Instead, set the unused variable to a blank string:

```
function bolg_preprocess_page (&$variables) {
  // From the Zen theme
  // Don't display empty help from node_help().
  if ($variables['help'] == "<div class=\"help\"><p></p>\n</div>") {
    $variables['help'] = '';
  }
}
```

In addition to the techniques you will encounter later in this chapter, much can be gleaned from other themes. Download and examine a variety of themes to see how other people have customized their page templates by adding, and modifying, their template variables.

Navigation and Menus

Your page template includes two variables containing navigation menus that you can place anywhere you like in your Web page: `$primary_links` and `$secondary_links`. These variables contain items from the two Drupal menus of the same name—primary and secondary links. Drupal menus are collections of links to both on-site and off-site URLs.

To add new items to the menus, you can use one of two methods:

- To add a link to an existing node, navigate to the editing screen for the node and adjust its menu settings as in Figure 4.3.
- You may also use the menu administration system to add a page to the menu as shown in Figure 4.4 by navigating to Administer, Site building, Menus, Add item. This method allows you to add links to off-site URLs.

To add subsection menu items, you use the same technique described above, but change the "Parent item" to the menu item in which your new subsection ought to be included. For example, suppose you have a set of primary links containing "Mammal," "Amphibian," and "Reptile." To place "Kitten" as a subsection of "Mammal," you would set the "Parent item" to be "Mammal" when adding the menu information for the "Kitten" node.

More menus into your page template

The menu module provides a block for every menu, and blocks can be placed into any region on the site. To display a menu in a block, navigate to Administer, Site building, Blocks. Complete the on-screen instructions to add the menu to a Web site region. More information about creating custom, task-based menus appears in Chapter 8.

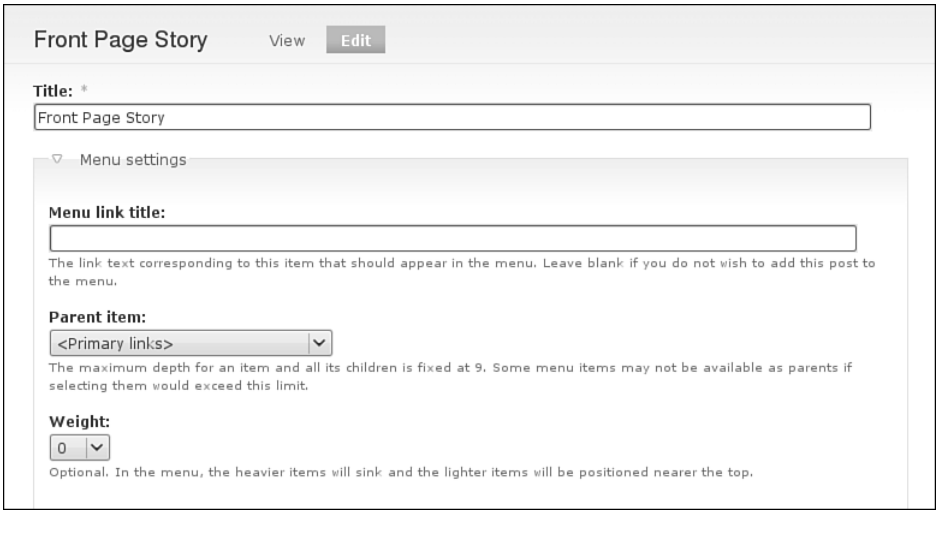


FIGURE 4.3 Adding a node to a menu from the node editing screen.

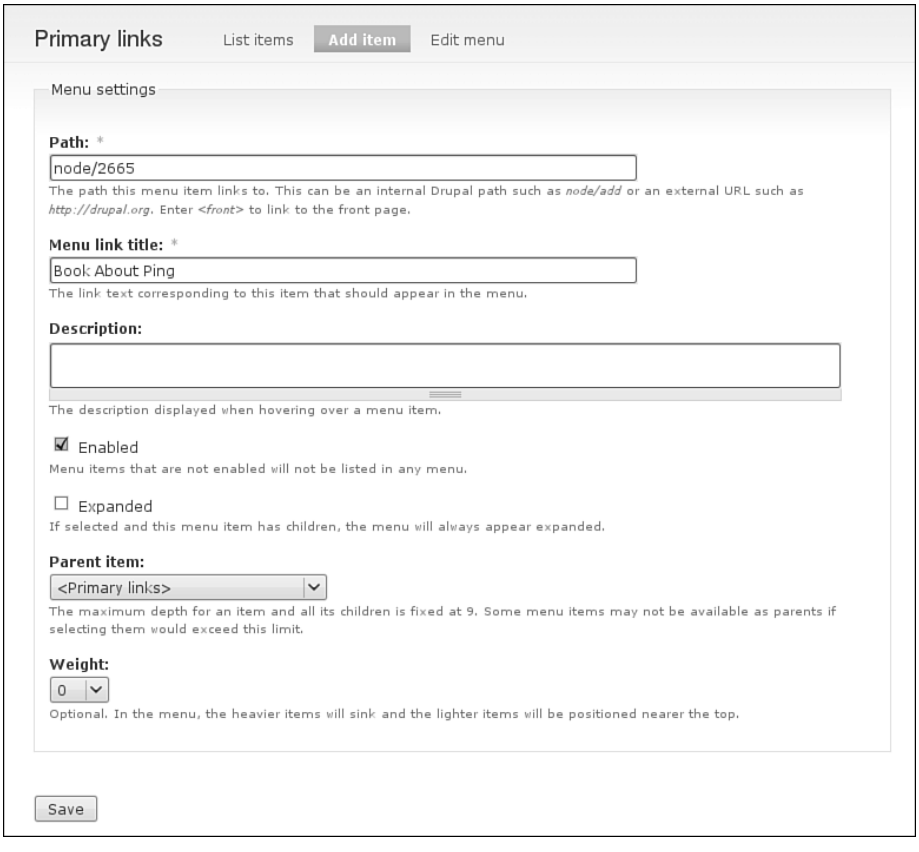


FIGURE 4.4 Adding a path to Primary links from the menu administration area.

Within the menu administration area, you can specify which menu is used for `$primary_links` and which menu is used for `$secondary_links`. By default, the variable `$primary_links` contains menu items from the menu "Primary links" and the variable `$secondary_links` contains items from the menu "Secondary links." To alter the menus that are used for these two navigation variables, navigate to Administer, Site building, Menus, Settings and adjust the settings as appropriate.

The variable `$secondary_links` can be configured in one of two ways: Either this menu can contain a second set of sitewide links for your site with "secondary" content (for example, legal notice, contact information), or you can configure `$secondary_links` to contain the relevant subsection navigation for your primary links. Use the following steps to change the default behavior:

1. Navigate to Administer, Site building, Menus.
2. Choose the Settings tab.
3. Change the "Source for the secondary links" so that it matches the menu that is set in the "Source for the primary links."
4. Scroll to the bottom of the Web page and click "Save configuration."

The page template variable `$secondary_links` now contains the subsection links that have been defined for each of the items in `$primary_links`. Referring to the previous example, "Kitten" will now be displayed in the output of `$secondary_links` when you select "Mammal" from the list of menu options provided by the variable `$primary_links`.

Theming Menus

A menu is built from three nested parts: the menu tree, the menu items (the "leaves" on the menu tree), and the menu item links. It is possible to alter the HTML for each of these components, although in most cases customizing the CSS for the default XHTML markup will be enough to make your menus look great. In addition to their basic structure, menus contain information about the menu leaves. For example, Figure 4.5 shows the active trail of the current page, Modules, and includes a menu of items that are collapsed, and expanded.

Depending on the type of menu items you want to alter, there are two relevant strategies:

- To alter the contents of the variables `$primary_links` and `$secondary_links`, use the page's preprocess function.
- To alter the markup for all menus, use theme functions.

Drop-down menus

The variables `$primary_links` and `$secondary_links` contain only the top-level menu items for their respective menus. If you would like to use a tree-like structure (useful for drop-down or fly-out menus) for your primary or secondary links, you must use the block version of your menu instead of the theme variables. The modules MenuTree and Nice Menus both create drop-down menus from your navigation variables. The project pages for these two modules can be found at `http://drupal.org/project/menutree` and `http://drupal.org/project/nice_menus`, respectively. Compare their features and choose the most appropriate module for your needs.

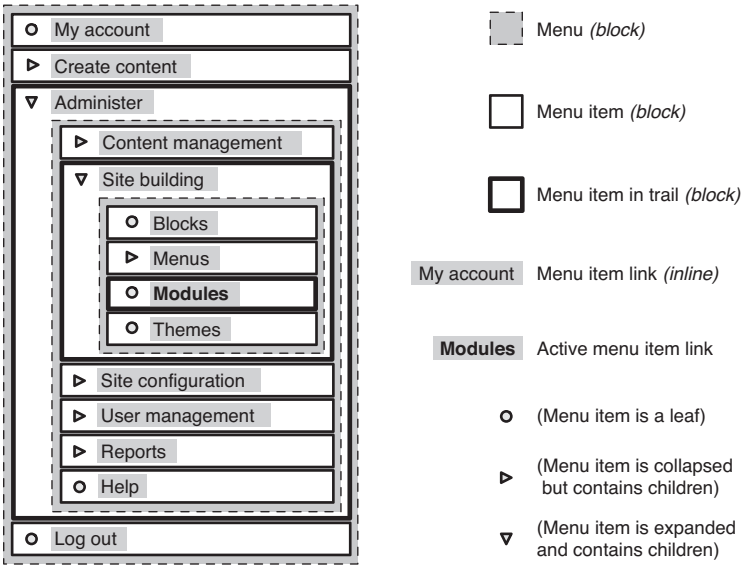


FIGURE 4.5 Menus are built of a menu tree, the menu items, and menu item links.

The primary and secondary links are registered theme variables. You may alter their contents by using the page's preprocess function. The variables themselves consist of an array of links and attributes. To make changes, you must loop through the list of links and alter each one individually. For example, if you decide to add a new class to each menu item that is related to its position in the menu, you could use the code snippet below. This technique would be useful if you wanted to add an icon to each menu item, because it relies on the exact order of the menu items. Once this order is set, you may not alter the order of the menu items without also updating the corresponding CSS styles.

```
function bolg_preprocess_page(&$variables) {
// Make a shortcut for the primary links variables
$primary_links = $variables['primary_links'];

// Loop through the menu, adding a new class for CSS selections
$i = 1;

foreach ($primary_links as $link => $attributes) {
  // Append the new class to existing classes for each menu item
```

```
  $class = $attributes['attributes']['class'] . " item-$i";

  // Add revised classes back to the primary links temp variable
  $primary_links[$link]['attributes']['class'] = $class;
  $i++;
}
 // End of the foreach loop

// reset the variable to contain the new markup
$variables['primary_links'] = $primary_links;

} // End of the preprocess function
```

Using the appropriate unique identifier for the primary links, add the new classes to your style sheet:

```
#primary_links .item-1 { /* styles for the first menu item */ }
```

This technique works well if you want to add styles based on the order of options in a menu. Menus are stored in an associative array and have a unique key assigned to each item. To create a unique menu item identifier, replace the variable `$i` with the variable `$link` in the snippet given earlier. Your menu items will now be assigned a unique identifier that does not change even when the order of the menu items is altered.

For more information about how menus are constructed and themed, read the API documentation at `http://api.drupal.org/api/function/theme_links/6` and `http://api.drupal.org/api/group/menu/6` (scroll to the list of theme functions).

Grid Work

In Chapter 1 of this book, you read about Web page design and were introduced to "regions" within a page template. Now you are ready to define the regions within your own page template and to then insert information into these defined spaces. There is no limit on how large or small a region can be within your page template. You may choose to stack many blocks into a region, or you may prefer to have only one block contained in a region. Figure 4.6 shows five of the regions available in the Zen theme

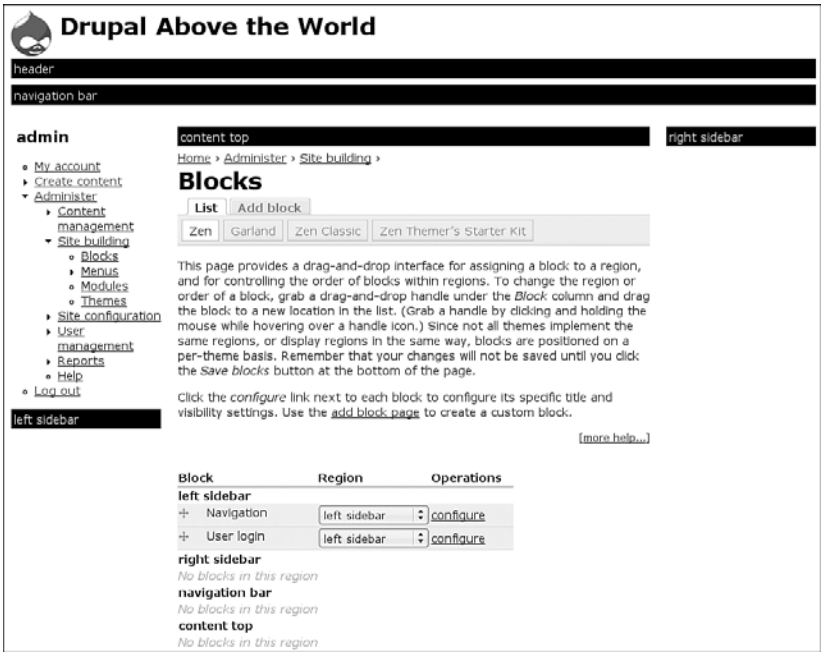


FIGURE 4.6 Five regions in the Zen theme, each with a different position and size.

as black bars. As you can see, the sizes of these regions differ depending on their location in the page.

Regions

Regions are used to place Drupal "blocks" into a Web site. These blocks may include site navigation menus, custom views, module tools, or custom PHP snippets. To see a list of the blocks that are currently available for your site, navigate to Administer, Site building, Blocks. Figure 4.7 shows the blocks that are available for the Hear the North site. This Web site has only a few modules installed, including a newsletter management tool Simplenews.

You can adjust the placement of these blocks by dragging and dropping the crosshair icon to a new region. To enable disabled blocks, drag them to a new region. To disable blocks, drag them back to the "Disabled" section. After updating the placement of blocks, you must click the button "Save blocks" to commit your changes to the

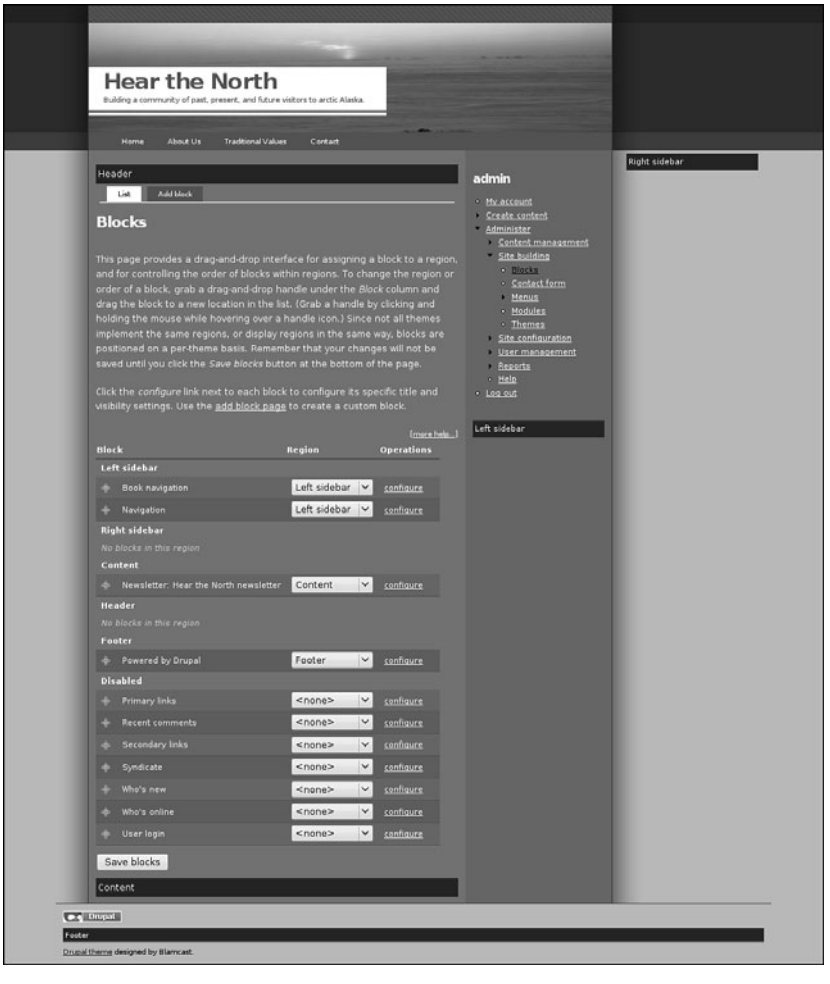


FIGURE 4.7 Blocks available on the Hear the North Web site.

database. You may also change the order of several blocks within a region using the same technique.

Adding a new region to your template is a multistep process:

1. Edit your theme's info file and add the regions as follows:

```
regions[new_region_name] = Human-readable region name
regions[second_region_name] = Another region name
```

2. Edit the file `page.tpl.php` and print your new regions to the structure of your page. Use the variable names you established in your theme's info file.

```
<?php print $new_region_name ?>
```

3. Clear the cache to reset the theme registry and enable the new regions. Navigate to Administer, Site configuration, Performance. Scroll to the bottom of the Web page and click "Clear cached data."
4. You should now be able to place blocks into your new regions by navigating to Administer, Site building, Blocks.

Here is the basic page template repeated from Chapter 3. A few changes have been made including the inclusion of new HTML divisions and one new region (marked in bold) that can be positioned with CSS. Putting these regions after the main content of the site will make the content appear more important to search engines, thereby increasing its rank in search engine results.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
   lang="<?php print $language->language ?>"
   xml:lang="<?php print $language->language ?>">
<head>
   <title><?php print $head_title; ?></title>
   <?php print $head; ?>
   <?php print $styles; ?>
   <?php print $scripts; ?>
</head>

<body class="<?php print $body_classes ?>">
<div id="main">
   <div id="page_title"><?php print $title; ?></div>
   <div id="utils-help"><?php print $help; ?></div>
   <div id="utils-messages"><?php print $messages; ?></div>
   <div id="utils-tab"><?php print $tabs; ?></div>
   <div id="main_content"><?php print $content; ?></div>
   <div id="utils-rss"><?php print $feed_icons; ?></div>
```

```
<div id="new-region-name"><?php print $new-region-name; ?></div>
</div>

<div id="sidebar-left"><?php print $left; ?></div>
<div id="sidebar-right"><?php print $right; ?></div>

<div id="footer"><?php print $region_footer; ?></div>
<?php print $closure; ?>
</body>
</html>
```

Blocks

With your regions established, you can now fill them with blocks. Blocks may be generated by Drupal core modules, contributed modules, or custom PHP snippets, including lists of content created by the Views module. For more information on creating a custom view, refer to Chapter 2.

Commonly used blocks include the following:

- Navigation menus (created in Administer, Site building, Menus)
- Lists of content (Views module; see Chapter 2)
- Login forms (Drupal core; turned on by default)
- Site categories (Drupal's Taxonomy module)
- Recent comments (Drupal's Comment module)
- Search (Drupal's search module)
- Author information (Drupal's profile module)
- Five-star ratings (`http://drupal.org/project/fivestar`)
- Facebook, Digg, and social bookmarking links (`http://drupal.org/project/service_links`)
- Similar entries (`http://drupal.org/project/similar`)

You can also create custom blocks with text, images, and even your own snippets of PHP code. Sample PHP snippets are available from the Drupal Web site at `http://drupal.org/node/21867`. To create a custom block, follow these steps:

1. Navigate to Administer, Site Building, Modules and enable the PHP Filter module. You may also need to adjust the permissions for this input format by navigating to Administer, Site configuration, Input formats and clicking on the "configure" link next to PHP filter.
2. Navigate to Administer, Site building, Blocks.
3. Select the tab "Add block."
4. Add a "Block description." This description specifies how the block will be identified in the administration area and is a required field.
5. Add a "Block title" if you would like a title to appear at the top of the displayed block. This field is optional.
6. Put your text, images, and PHP snippet into the "Block body." You could also use plain text or HTML markup here if it was appropriate for your block.
7. Update the "Input format" to PHP.
8. Adjust the visibility settings for the "User," "Role," and "Page" roles.
9. Scroll to the bottom of the Web page and click "Save."

PHP snippets in blocks
Blocks with custom PHP snippets could break the display of your site if they contain errors. Be sure to carefully test your snippets before placing them into a block. Place your PHP snippet into the body of a private page to confirm that it will not break your site before deploying the snippet as a block.

Sites will sometimes have more screen real estate dedicated to blocks than to the main content on each page, especially when the blocks provide additional information for the node that is displayed on the page, such as author profile information or related content. Don't be shy! Enable the most appropriate blocks for each part of your Web site. Blocks are included in Drupal's caching system and will not harm the overall performance of your site. To enable caching for blocks, navigate to Administer, Site configuration, Performance. Under the section "Block cache," choose "Enabled." Scroll to the bottom of the Web page and click "Save configuration."

Customizing the Markup of Blocks

You may change the markup of the blocks displayed in your page template by creating a new template file, `block.tpl.php`. Drupal's default for this template contains only a few wrapper HTML elements:

```
<div id="block-<?php print $block->module .'-'. $block->delta; ?>"
  class="block block-<?php print $block->module ?>">
<?php if ($block->subject) { ?>
  <h2><?php print $block->subject; ?></h2>
<?php } ?>

  <div class="content">
    <?php print $block->content ?>
  </div>
</div>
```

For blocks provided by Drupal core, the variable `$block->delta` represents the order in which this block was created. For example, the first block has a delta value of 1, the second has a delta value of 2, and so on. In rendered HTML, the first line would look like this:

```
<div id="block-user-1" class="block block-user">
```

As you can see, the output is not nearly as complicated as the variables would suggest! Check the output to see what your module is using for its delta value. Some modules provide a text delta instead of a numeric delta.

A full list of block template variables is available from the default block template. This file can be found in your Drupal system files: `modules/system/block.tpl.php`. A full list of the variables is also available online at `http://api.drupal.org/api/file/modules/system/block.tpl.php`.

Search

The default Drupal core engine comes with a module that allows you to search the contents of your site. There are four steps to enabling search on your site: enable the search module; update the permissions for users to search content; index the content on a regular basis through the use of a "cron job"; and display the search form to site visitors.

1. The Search module is not enabled by default. To enable this module, navigate to Administer, Site Building, Modules; enable the module by placing a check mark next to it, scrolling to the bottom of the Web page, and clicking "Save."

2. Next you must enable the permissions for the appropriate roles in your site. Navigate to Administer, User Management, Permissions. To enable searching for all users, make sure "search content" and "advanced search" are enabled for "anonymous user."
3. Drupal's search module does not search the content of the database directly because this operation would be too time-consuming. Instead, it searches an index of your content (similar to an index at the back of a book). To initiate this process of creating or updating the index, navigate to Administer, Reports, Status report. Click on the link "run cron manually." The page will automatically refresh, showing you the cron maintenance task that was last run "less than a few seconds ago." For more information on configuring cron jobs for Drupal, refer to Chapter 2.
4. Two styles of search tools are available for Drupal themes; Figure 4.8 compares these two search forms. On the left side of the screen, the top option is the theme's search box (which has no heading); the second option is the Search form block (which has a heading). If you like, you can customize the Search

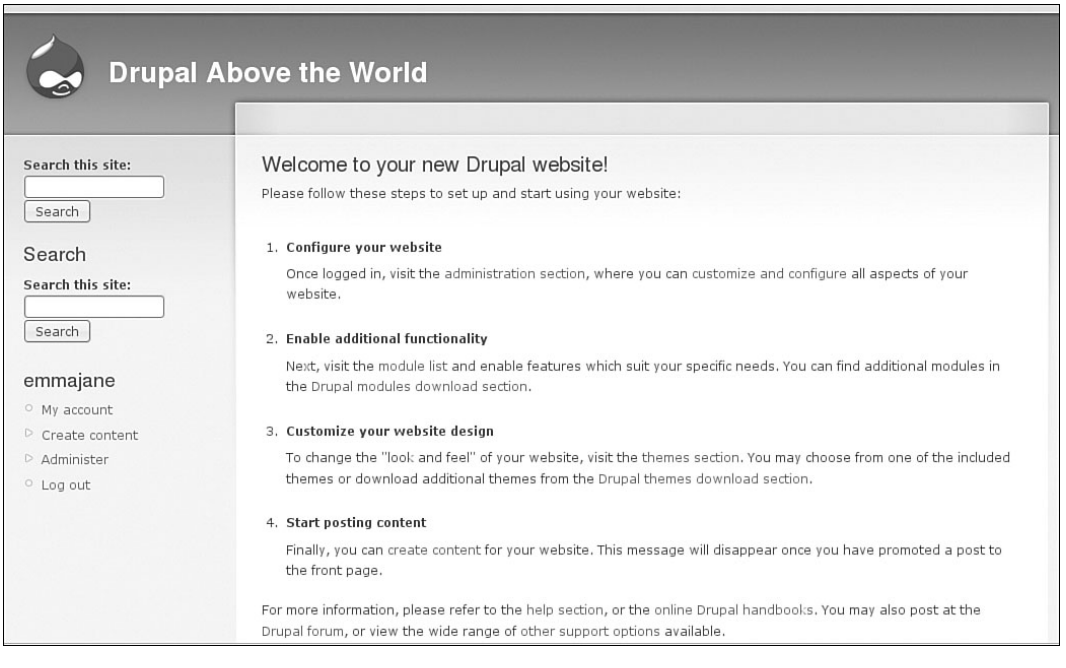


FIGURE 4.8 There are two ways to enable a search box in a Drupal theme. On the left side of the screen, the top option is the theme's search form; the bottom option is the Search form block.

form block to remove the heading. Although these two search forms have a very similar appearance, they are actually applied in quite different ways. The Search form block may be placed only into an existing region; in contrast, the theme's search box may be placed anywhere within the page template.

To enable the theme's search box, add the following PHP snippet to your theme's `page.tpl.php` file at the appropriate location:

```
<?php print $search_box ?>
```

To enable the search block, use these steps:

1. Navigate to Administer, Site Building, Blocks.
2. Scroll down to the "Disabled" section.
3. Select a region for the search form from the select menu.
4. Scroll to the bottom of the Web page and click "Save blocks."

Your search box should now appear as a block within your Web site. To further customize the options for the search block, you can navigate to Administer, Site building, Blocks and click on the "configure" link next to the Search form.

Changing Templates

In this chapter you have learned how to create a template for your page and how to customize the page elements. In this section you will see how to change the page templates that are used for different sections of your Web site. You may want to use different templates for each of the following tasks and types of pages:

- Editing content
- Displaying a content type
- User login
- Front page
- Categories
- Offline or maintenance page

Some of these templates are provided by default; others you will need to build from scratch. The online documentation has a complete list of all default templates provided by Drupal at `http://drupal.org/node/190815`. This section describes several of the page-specific template options.

Assigning themes to different parts of your site

This section describes how to change the template that is used within a single theme. If you need to assign whole themes to different parts of your Web site, you will need a more powerful toolkit. The contributed module known as sections will allow you to do exactly this. For more information about this project, visit `http://drupal.org/project/sections`.

If you need to provide even more customization on a per-section basis, you may need The Organic Groups module. This module enables authorized users to create and manage their own "groups." Each group gets its own theme, language, and taxonomy. The techniques described in this book could be applied to each theme for each group on the Web site. For more information about this project, visit `http://drupal.org/project/og`.

Custom Front Page

What if you need a front page that has more—or fewer—regions than are provided by a certain template? What if the front page needs to have a bigger banner and a smaller content area? What if you need to make so many changes that it feels like the front page needs a theme all of its own? Fortunately, it is very easy to create a custom front page template for your Drupal site—so easy, in fact, that it is difficult to fill up a whole section of this chapter with information about making a new front page template!

To make a custom front page template, follow these steps:

1. Create a new page template file with the name `page-front.tpl.php`. This is a special file name recognized by Drupal as being a unique template to be used on only the front page of the Web site.
2. Clear the theme registry by navigating to Administer, Site configuration, Performance; scroll to the bottom of the Web page and click "clear cached data."
3. Navigate to the front page of your Web site and marvel!

All pages other than the front page will still use the template file `page.tpl.php` (unless additional page-specific templates are used elsewhere in the site).

Using a view on the front page

If you are using the Views module, you can use the page view to create a custom front page. Once you have created the view and assigned an alias to it, navigate to Administer, Site configuration, Site information. Scroll to the bottom of the Web page and adjust the setting for the "Default front page" so that it uses the new view page alias for the default front page.

Custom Offline Page

Unfortunately, bad things sometimes happen to good Web sites, and the Web sites have to go offline. Drupal provides a default template when a connection cannot be made to the database. In addition, the site can be directed to enter "maintenance" mode so that you can perform some upgrades or other feature enhancements. Figure 4.9 and Figure 4.10 show the default templates for these two offline pages.

The offline message template will appear only to visitors who are not authenticated; administrators will still have access to the Web site as they perform their upgrades when a site is "under maintenance." To customize these pages, complete the following steps:

1. Copy the default maintenance page from the Drupal core directory `modules/system/maintenance-page.tpl.php` to your theme's directory.
2. Make a second copy of the file for the offline template and name it `maintenance-page-offline.tpl.php`.
3. You should now have two new files in your theme's directory:
   - `maintenance-page.tpl.php`: "maintenance" mode
   - `maintenance-page-offline.tpl.php`: "database is offline"
4. Adjust these two new templates to suit your needs.
5. Open your site's configuration file in a text editor. (This file is found in `sites/yourdomainname.com/settings.php`. It is not a theme file, and it is probably write-protected.)
6. Remove the `#` symbol from the following lines:
   - Line 173: `# $conf = array(`
   - Line 175: `#   'theme_default' => 'your_theme_name'`
   - Line 187: `#   'maintenance_theme' => 'your_theme_name'`
   - Line 214: `# );`
7. Save the changes and make the file read-only again.

The next time you put your Web site into maintenance mode (or if your database server ever goes offline), you will be able to show your customized apology to the world instead of the default Drupal "maintenance" message.

FIGURE 4.9 Offline message for site "under maintenance."

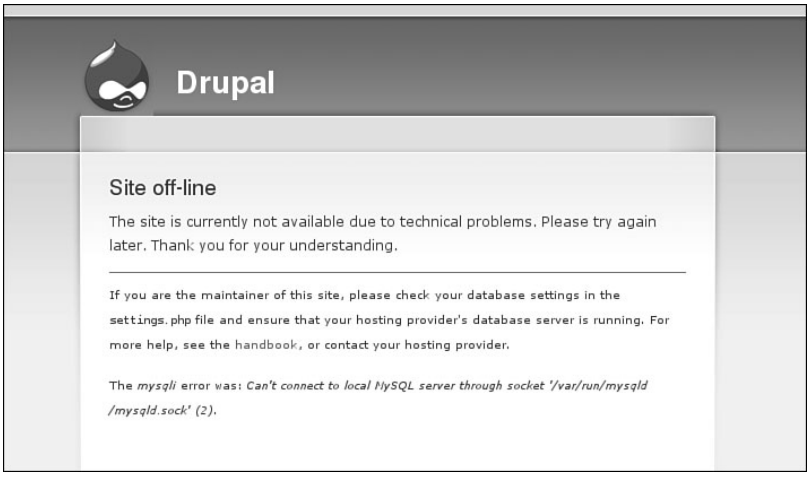


FIGURE 4.10 Offline message when the database connection fails.

In the section "Custom Front Page," you learned how to create a custom front page template. The template was activated when the current page was the front page of the Web site. You may take advantage of this technique to target other types of pages as well. Page templates are activated according to Drupal's internal path for the current page.

Internal path and URL alias
This technique works only with the internal path for a page. You cannot use URL aliases. You will learn how to work with aliases in the next section. For now, you may only use paths that are related to Drupal core terminology. For example, `node/5 and node/5/edit` are both internal paths that can be tied to a specific page template, whereas `books/fiction/story-about-ping` is a URL alias. Use the Devel module to obtain a list of suggested template files for each page. If none of the suggested templates matches your needs, consider skipping ahead to the next section to discover alternative ways to create template files.

Drupal looks through a list of suggested templates from most specific to least specific and checks your theme's directory for a matching template file. Once it finds a template that matches the criteria, it applies that template to the page. The following list gives examples of the templates that would match for each of the pages:

- `http://www.example.com/node/5`
    - `page-node-5.tpl.php`
    - `page-node.tpl.php`
    - `page.tpl.php`
- `http://www.example.com/node/5/edit`
    - `page-node-edit.tpl.php`
    - `page-node-5.tpl.php`
    - `page-node.tpl.php`
    - `page.tpl.php`
- `http://www.example.com/admin/build/block`
    - `page-admin-build-block.tpl.php`
    - `page-admin-build.tpl.php`
    - `page-admin.tpl.php`

    - `page.tpl.php`
- `http://www.example.com/books/fiction/story-about-ping`
    - `page-node-2665.tpl.php`
    - `page-node.tpl.php`
    - `page.tpl.php`

The last item in the list is using a URL alias. There is not a single template that can be used by default to match any of the words in the URL to assign a template. Instead, you must know the exact node ID for the page to find a node-specific template match. These template suggestions exist automatically, so use them whenever you need to create a new template with the same file name and then theme it according to your needs. You may also need to clear the theme registry to see your new template in action.

Alias: Page

Do you remember the TV show *Alias?* It was full of wigs and disguises and trickery and deception and intrigue! URL aliases are a bit like throwing a wig onto a system path—they change the way the path looks, but keep the content the same. If you want your site to use URLs that are more closely tied to page content than `node/2868`, you will need to use the module Path to create URL aliases. The bad news is that Drupal's theming system cannot recognize the URL aliases that you have created with the wigs and the dark sunglasses. Instead, you must explicitly show Drupal how you want to convert these URLs into a template suggestion. In the next section, you will learn how to further customize this process to create template suggestions for each category.

Template overload
Do you really need a whole new page template? Think carefully before implementing the ideas presented in this section. For each new page template you create, you will need to maintain the markup for an entirely new page. The more you add, the more you have to maintain. There may be other, less time-intensive ways to simplify a layout—for example, displaying blocks only on certain pages.

The first step in this process is to grab the URL and examine its components before the page template is processed. Using the URL alias, you will compile a new list of suggested page templates. Being careful to match the alias for the page you want to

redesign, you then add a new template file to your theme. Now when Drupal looks for the best match for its page template, it will use your new list of suggested file names and find the new page template.

New Templates from Aliased URLs

The work of compiling the new list of suggested templates happens in the page preprocess function in your theme's `template.php` file. If you have already created a `preprocess_page` function in your theme's `template.php` file, you may add this snippet to either the beginning or the end of the function. If you do not already have this function, you will need to include the very first (and very last) lines of this snippet in your theme's `template.php` file.

It takes several steps to compile a new list of suggested templates for the URL alias of your page:

1. Confirm that the module `path` is enabled. Without this module, your site will not have URL aliases and this function will be irrelevant.
2. By default, Drupal allows you to access the system path, but not the URL alias. You need to use a special decoder ring, `drupal_get_path_alias`, to convert the system path back to its URL alias.
3. Break the URL alias into its components using PHP function `explode`. You will use these components to build the new page template file name.
4. Make sure your Web page is not an editing page. If it is, Drupal's templates can be used and this function becomes irrelevant.
5. Create a variable to hold the new template suggestions, and establish the base word for the new template's file names. You could use any word here, but using the base word "page" allows you to keep all page templates together. For example, `page-your-custom-url.tpl.php` would be alphabetically close to `page-front.tpl.php`.
6. Loop through each part of the URL and build new template suggestions. This mimics the way Drupal offers its templates. For example, if your URL alias is `books/fiction/story-about-ping`, you will now be able to create three new page templates: `page-books.tpl.php`, `page-books-fiction.tpl.php`, and `page-books-fiction-story-about-ping.tpl.php`.
7. Add the new template suggestion to a list that will be handed back to Drupal.
8. Finally, return the list of suggested template names back to Drupal.

In your file `template.php`, the PHP snippet for these eight steps is as follows:

```
function bolg_preprocess_page(&$variables) {
// Step 1:
if (module_exists('path')) {

// Step 2:
$path_alias = drupal_get_path_alias($_GET['q']);

// Step 3:
$alias_parts = explode('/', $path_alias);

// Step 4:
$last = array_reverse($alias_parts);
$last_part = $last[0];
if ($last_part != "edit") {

// Step 5:
  $templates = array();
  $template_name = "page";

// Step 6:
  foreach ($alias_parts as $part) {
    $template_name = $template_name . '-' . $part;

// Step 7:
    $templates[] = $template_name;
  }

// Step 8:
  $variables['template_files'] = $templates;

} // End of the edit check
} // End of the check for the path module
} // End of the preprocess_page function
```

After you place this snippet in your theme's `template.php` file, you may use any part of the URL alias as a page template name. Note, however, that you must refresh the theme registry before Drupal sees your new template suggestions.

Page Templates for Views

The Views module is very clever. When you provide a URL alias for your page view, it automatically performs its version of the function that was described in the previous section. For example, if you have a view with the URL alias `recent/screencasts`, the Views module will automatically generate the following page template suggestions: `page-recent.tpl.php` and `page-recent-screencasts.tpl.php`. The default page template, `page.tpl.php`, will be used there if none of these files exist within the theme's directory.

Adding CSS Classes

The Zen theme allows designers to adapt their layout based on the classes that are applied to the body. You can add this level of customization to your theme as well. To add classes to your page, you will need to alter the contents of the page variable `$body_classes`. This variable contains a list of classes all separated by a space. To add new classes to this variable, you can use the same function that was described previously in this chapter. In the code outlined in the section "New Templates from Aliased URLs," replace step 8 with the following lines (the first line is a comment, not part of the functioning code):

```
// Step 8:
$classes = implode(' page-', $templates);
$variables['body_classes'] = $variables['body_classes'] . ' $classes';
```

This will add your new body classes to the end of the list of default classes.

Additional body classes are available

If you want to have even more classes available for theming, you may find the Themer module useful. This tiny module creates a suite of CSS classes that can be applied throughout your theme. Additional information is available on the project page at `http://drupal.org/project/themer`.

Page Templates for Content Types

If necessary, you can change the way a node is displayed within a page with Drupal's node templates. If you knew that one of your content types needed a different page layout, however, you could assign a new page template to that content type. This process is almost the same as that followed in the previous examples.

To make a content type-specific page template, you will need to know which type of content you are looking at. The only time you can know this with certainty is when you are looking at a page that contains only one node. This page would normally use the page template `page-node.tpl.php`.

To create a template suggestion based on content type, you will need to replace steps 6, 7, and 8 of the preprocess function described in the section "New Templates from Aliased URLs" with the following snippet. Notice the use of `arg()` in this example; `arg()` is a special variable that grabs individual parameters from the system path for the displayed page. For example, the value of `arg(0)` for `node/2868` is "node" and the value of `arg(1)` is 2868.

```
if (arg(0) == "node" && is_numeric(arg(1))) {
   $node_type = $variables['node']->type;
   $variables['template_files'] = "$template_name-node-$node_type.tpl.php";
}
```

If you want to make templates for both URL aliases and content types, you can add this snippet *after* step 8 in the code snippet described in "New Templates from Aliased URLs:"

```
if (arg(0) == "node" && is_numeric(arg(1))) {
   $node_type = $variables['node']->type;
   array_push($variables['template_files'], "$template_name-node-$node_type.tpl.
php");
}
```

The examples in this section should give you a solid toolkit for creating unique page templates. You may think of even more ways to customize your templates, too!

Taxonomy Templates

The previous section described how to build new templates based on URL aliases and content type. When you are designing a site to have category-specific enhancements, it is very likely that you want to change the colors or graphical elements of the page template. This section explores ways to create a new page template so as to add color-specific sections and new variables. To accomplish this feat, you will use the same techniques you learned in the previous section.

Unfortunately, categories are easily edited and are not associated with permanent machine names. You may find it helpful to print the taxonomy variable to the page to see how categories are stored and accessed. You can also obtain this information by using the developer module Themer Info tool in the Devel module. Refer to Chapter 2 for more information on using this module.

Here are the contents of one taxonomy variable:

```
[taxonomy] => Array
  (
     [3] => stdClass Object
        (
          [tid] => 3
          [vid] => 1
          [name] => Available for retail and wholesale.
          [description] =>
          [weight] => 0
         )

      [11] => stdClass Object
         (
           [tid] => 11
           [vid] => 2
           [name] => Books Published by The Ginger Press
           [description] =>
           [weight] => 0
         )
   )
```

In this example, the category being used to change the template variable is the first category contained in the array of data in the taxonomy variable. The first four steps of the `preprocess_page` function described in "New Templates from Aliased URLs" section are repeated. At this point, you should adjust the variable `$target_tax` so that it matches the position of the category you want to use to distinguish between sections on your site. This function assumes that you are working within one vocabulary and that each term is a different template. You will need to adjust the scripting if your site differs from this model.

The explanations of steps 1 through 4 can be found in the section "New Templates from Aliased URLs." The new steps perform the following actions:

5. Check whether this page has a system path of `node/nid`. This snippet will work only if you are displaying a single node of any content type.
6. Check whether this page has been assigned a category. Retrieve the whole array of categories if it does.
7. Retrieve the name of the category.
8. Convert the category name to a plain text string of characters suitable for a file name. This operation includes replacing spaces with a dash and converting all characters to lowercase.
9. Add the new template suggestion to the list of page template suggestions; add the category name to the list of existing body classes.

```
function bolg_preprocess_page(&$variables) {
// Step 1:
if (module_exists('path')) {

// Step 2:
$url_alias = drupal_get_path_alias($_GET['q']);

// Step 3:
$alias_parts = explode('/', $url_alias);

// Step 4:
$last = array_reverse($alias_parts);
$last_part = $last[0];
if ($last_part != "edit") {
```

```
// Step 5:
if (arg(0) == "node" && is_numeric(arg(1))) {

// Step 6:
if (isset($variables['node']->taxonomy)) {
   $target_tax = 0;
   $node_tax = $variables['node']->taxonomy;

// Step 7:
   $tid = array_keys($node_tax);
   $name = $node_tax[$tid[$target_tax]]->name;

// Step 8:
   $clean_name = check_plain($name);
   $dash_name = str_replace(" ", "-", $clean_name);
   $lc_name = strtolower($dash_name);

// Step 9:
array_push($variables['template_files'], "page-tax-$lc_name.tpl.php");
$variables['body_classes'] .= $variables['body_classes'] . " tax-$lc_name";

} // End of the taxonomy check
} // End of the node/nid check
} // End of the edit check
} // End of the check for the path module
} // End of the preprocess_page function
```

Graphical Headers

The last function introduced in this chapter allows you to change the template or add a new CSS class to a page based on the category assigned to a page. Wouldn't it be neat if you could change the graphical header for that page as well? With the snippet of code provided here, you will be able to place images into a folder in your theme directory and have them be automatically displayed for unique categories within your Web site.

This snippet can be used as a replacement for step 9 in the preceding section, or it can be used as a further enhancement. It assumes that all of the images reside in a subdirectory of your theme named `tax` and that all image files are named with the lowercase extension `jpg`. You may change these settings, if necessary. The image files should all be named according to the following convention: Using the term name, replace all spaces with a dash and convert all letters to lowercase. A default image should also be available if a matching taxonomy-specific image cannot be found.

```
$image_dir = "tax";
$ext = "jpg";
$default_image_file = "FILENAME.jpg";

$image_dir = drupal_get_path('theme', 'bolg') . "/$image_dir");
$default_image = "$image_dir/$default_image_file";
$image = "$image_dir/$lc_name.$ext";

if (file_exists($image){
   $variables['tax_header'] = theme('image', $image, $clean_name, $clean_name);
} elseif (file_exists($default_image){
   $variables['tax_header'] = theme('image', $default_image, $clean_name, $clean_
name);
} else {
   $variables['tax_header'] = "";
}
```

Remember to put the default header graphic into the appropriate image folder in your theme!

Delivering Plain Content

Sometimes a stripped-down version of your site is more appropriate than one cluttered with bells and whistles. For example, "simpler is better" when you are aiming to provide a print-friendly version of a page or a mobile-friendly version of your Web site.

Print-Friendly Pages

There are two ways to prepare pages for printing. The first is to prepare a unique style sheet for printers. The browser will automatically detect style sheets that have been marked with a media type of "print" and format the page according to the print rules that have been specified. The second method uses a contributed module, Print, to enable links that direct the site visitor to new pages that use a print-friendly template.

CSS Print-Friendly Pages

Cascading Style Sheets (CSS) specify the media type they are targeting. When a page is displayed in a Web browser, you are viewing the styles that have been assigned to the page by the media types "all" and "screen." Eight other media types are available, including "print," "braille," "handheld," and "tv." A full list of media types is available from `http://www.w3.org/TR/CSS2/media.html#media-types`.

The "print" media type specifies how a page should be formatted when it is printed. Figure 4.11 shows a Web page formatted by a "screen" style sheet; Figure 4.12 shows the "print preview" for the same page. Parts of the page that are not relevant to the content being displayed have been eliminated. The elements that have been removed include the header, navigation elements, and quotes in the footer.

Most of the work in creating a print-friendly style sheet focuses on finding regions that can be "hidden" from view. To remove these variables from the print-friendly version of the page, the CSS property and attribute `display: none;` are used. The site name (HICK Tech) is also pulled into the display by using the property and attribute `display: block;`. To add a print-friendly style sheet to your site, you must register the new file in your theme's `.info` file and clear the theme registry by navigating to Administer, Site configuration, Performance; scrolling to the bottom of the Web page; and clicking "clear cached data." A print-specific CSS file is typically named `print.css`; however, there is no absolute requirement to use this file name. Set the print style sheet with the following snippet in your theme's `.info` file:

```
stylesheets[print][] = printstylesheet.css
```

The print style sheet for the HICK Tech Web site contains only the following styles:

```
/* Hide all information that is not unique content for this page */
#header-wrapper, #primary-links, #banner-image, .sidebar-right .sidebar-right,
```

FIGURE 4.11 HICK Tech Web site as it is displayed in a Web browser.

```
.breadcrumb, ul.primary, div.links, #bottomboxes, #footer {
        display: none;
}

/* The site name is set to "display: none"
   in the main style sheet, display it now*/
#print-sitename { display: block; }

/* Use print-friendly fonts */
body {
```

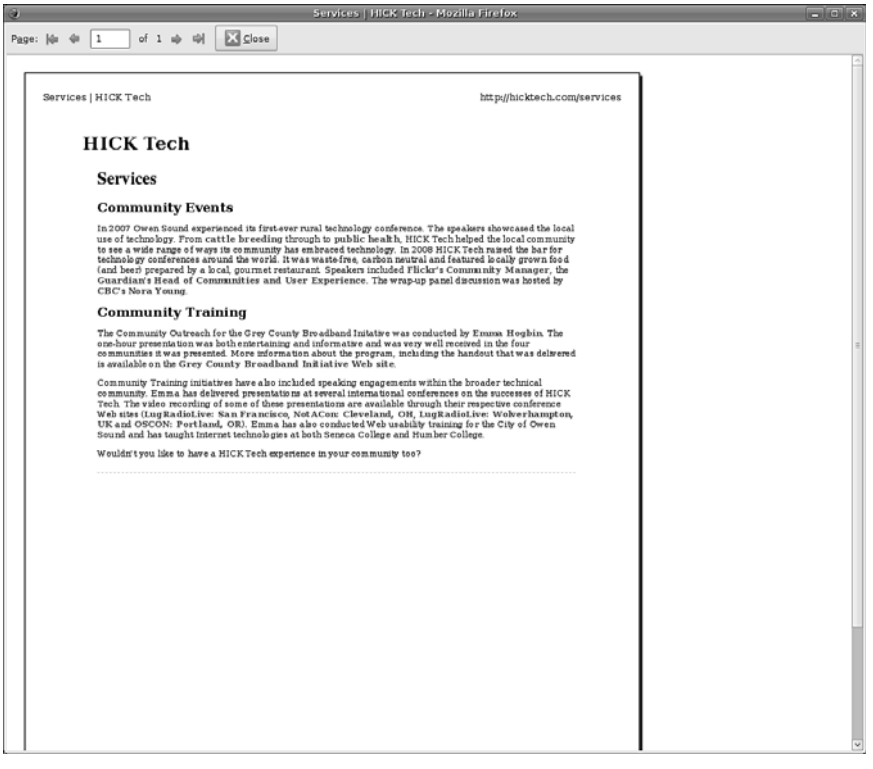


FIGURE 4.12 HICK Tech Web site seen in "print preview" mode using the print style sheet.

```
        font-family: Serif;
        color: #000;
        font-size: 1em;
        text-align: left;
}

/* Make sure the page is white, with no border, and properly aligned */
#wrapper {
        background: #fff;
        border: none;
        margin: 0;
        width: 100%;
}
```

To add your logo to the site name, you could place a background image on the site with the following CSS snippet:

```
#print-sitename {
      display: block;
      background-image: url(/path/to/the/image.gif);
}
```

If you are concerned about exact color matching (saving your visitor's valuable color ink cartridges), consider using a black-and-white logo here instead of your colored logo.

Several Drupal themes provide print-friendly CSS, including the default theme, Garland. Review the following themes for additional examples on how to create a print-friendly style sheet for your theme:

- AD Redoable (`http://drupal.org/project/ad_redoable`)
- NoProb (`http://drupal.org/project/noprob`)
- Pluralism (`http://drupal.org/project/pluralism`)
- Zen (`http://drupal.org/project/zen`)

The A List Apart article titled "Going to Print" by Eric Meyer provides excellent information and strategies for creating print-friendly pages using only CSS. This article can be found at `http://alistapart.com/articles/goingtoprint/`.

Print-Friendly Templates

Sometimes your Web site visitors will simply not believe that a print-friendly page is waiting to greet them in the printer. They may have had too many bad experiences with Web sites that do not provide a print-friendly CSS, and they may not understand the mechanics of Web site construction well enough to know such a thing is even possible. The CrochetMe Web site shown in Figure 4.13 shows a link to a print-friendly page (displayed in Figure 4.14) with all cruft removed. To create custom templates for your content, you must generate new links to the end of each node, create new templates with stripped-down markup, and notify the theme about these new (nonstandard) template files. Sounds like a lot of work, eh?

Print module to the rescue! With this nifty little module, you can easily enable print-friendly, email-this-page, and PDF links to all of your pages. For more information

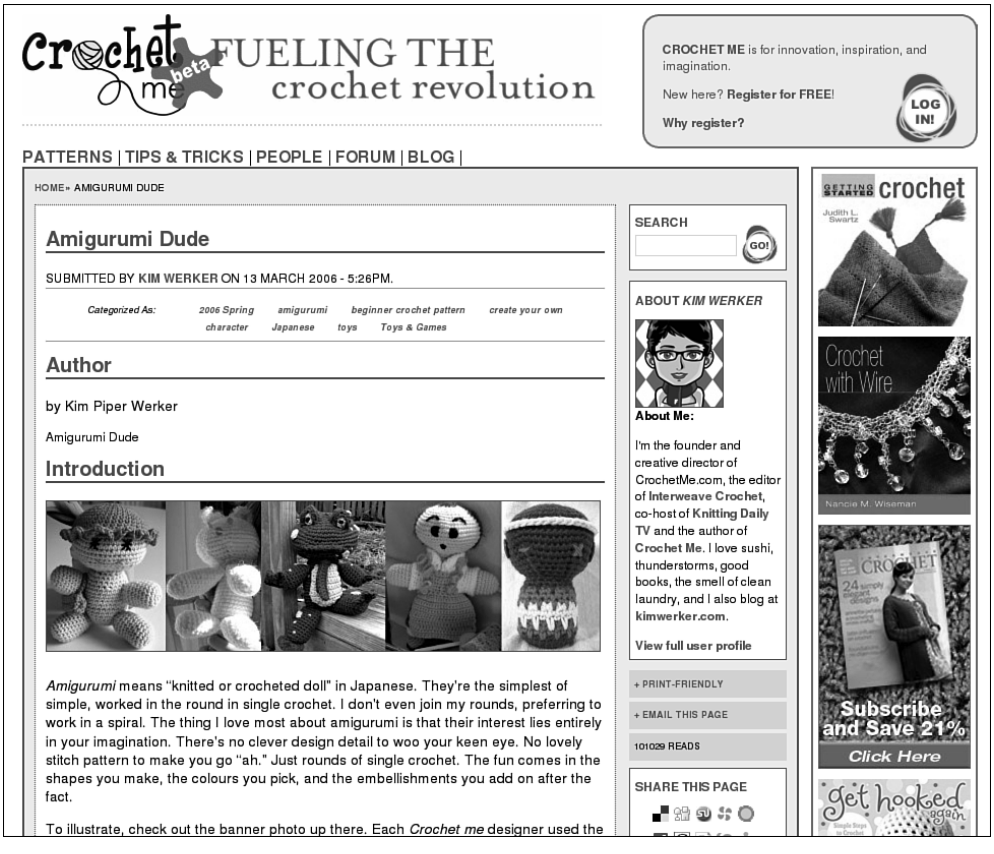
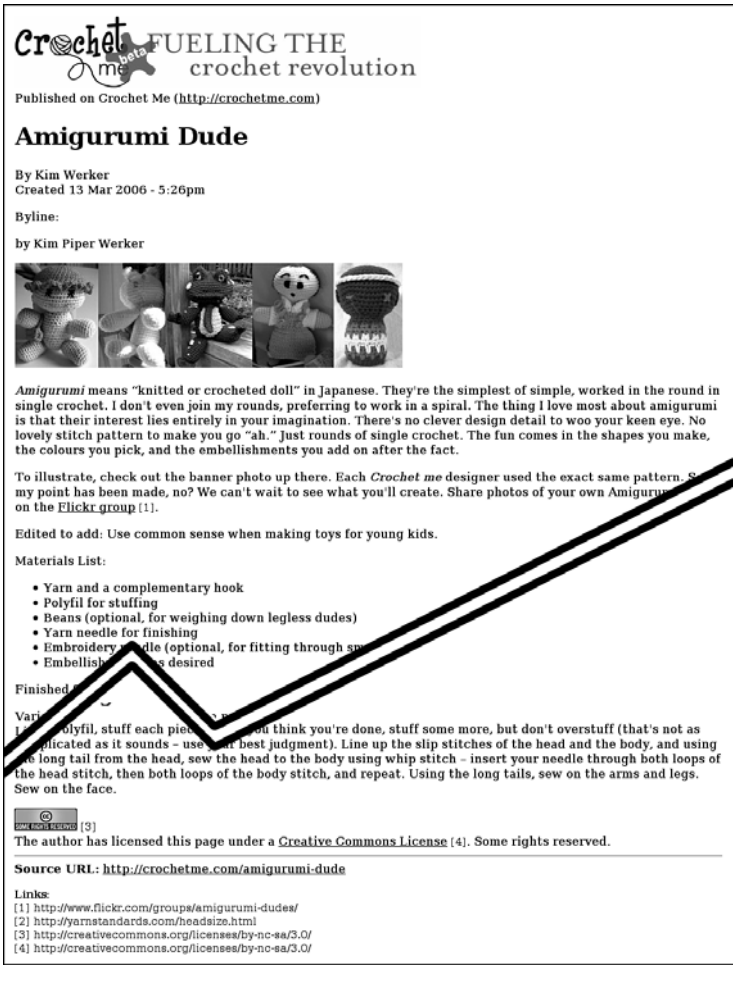


FIGURE 4.13 The CrochetMe Web site uses the Print module for its content. The links appears to the right of the content, below the author information.

about this module, and to download and install it, visit the module's project page at `http://drupal.org/project/print`.

Although this module does have the ability to create PDFs of pages, it requires a helper module. The recommended helper module, which is named `dompdf`, provides full CSS support and allows for excellent reproduction of the Web page. It does not, however, support Unicode character encoding or PDF headers. To install the `dompdf` module, you must install font support on your Web server. If you are not comfortable with system administration, or if you are using a shared hosting service, this functionality will be a little tricky to implement. For more information, visit the `dompdf` Web site at `http://www.digitaljunkies.ca/dompdf`.


Crochet me beta FUELING THE crochet revolution

Published on Crochet Me (http://crochetme.com)

Amigurumi Dude

By Kim Werker
Created 13 Mar 2006 - 5:26pm

Byline:

by Kim Piper Werker

Amigurumi means "knitted or crocheted doll" in Japanese. They're the simplest of simple, worked in the round in single crochet. I don't even join my rounds, preferring to work in a spiral. The thing I love most about amigurumi is that their interest lies entirely in your imagination. There's no clever design detail to woo your keen eye. No lovely stitch pattern to make you go "ah." Just rounds of single crochet. The fun comes in the shapes you make, the colours you pick, and the embellishments you add on after the fact.

To illustrate, check out the banner photo up there. Each *Crochet me* designer used the exact same pattern. S... my point has been made, no? We can't wait to see what you'll create. Share photos of your own Amigurum... on the Flickr group [1].

Edited to add: Use common sense when making toys for young kids.

Materials List:

- Yarn and a complementary hook
- Polyfil for stuffing
- Beans (optional, for weighing down legless dudes)
- Yarn needle for finishing
- Embroidery ...dle (optional, for fitting through sp...
- Embellish... ...s desired

Finished ...

Vari... ...

...olyfil, stuff each pie... ...u think you're done, stuff some more, but don't overstuff (that's not as ...ificated as it sounds – use ... best judgment). Line up the slip stitches of the head and the body, and using ...e long tail from the head, sew the head to the body using whip stitch – insert your needle through both loops of the head stitch, then both loops of the body stitch, and repeat. Using the long tails, sew on the arms and legs. Sew on the face.

SOME RIGHTS RESERVED [3]
The author has licensed this page under a Creative Commons License [4]. Some rights reserved.

Source URL: http://crochetme.com/amigurumi-dude

Links:
[1] http://www.flickr.com/groups/amigurumi-dudes/
[2] http://yarnstandards.com/headsize.html
[3] http://creativecommons.org/licenses/by-nc-sa/3.0/
[4] http://creativecommons.org/licenses/by-nc-sa/3.0/

FIGURE 4.14 Output of the Print module—a "print-friendly" page.

Mobile Devices

Handheld devices are becoming more common, to the point that having a site that can be navigated while "on the go" is a must for service-oriented businesses such as restaurants, shops, and social networking sites. If you do not have the resources to develop a mobile application, that does not mean you cannot provide a mobile-friendly version of your Web site. To provide this trimmed-down version of your site template, you may use the Mobile theme. This theme is intended to return only clean HTML with

no styling (although images embedded in your content are maintained). The links and sidebars are placed so that mobile or handheld devices can display the content first. For more information about this module, and to download and install it, visit the module's project page at `http://drupal.org/project/mobile`.

Once the Mobile theme is installed, you will still need to provide a URL for the mobile version of your Web site. To do so, complete the following steps:

1. Create a subdomain for the mobile version of your Web site. It is common practice to replace the "www" in your site's domain name with the letter "m."
2. Using the domain name you created in step 1, create a duplicate folder of your current site in Drupal's folder `sites`. For example, if you were adding a mobile version to the site `example.com`, the folder `sites` would include the following folders:
   - `example.com`
   - `m.example.com`

   These two folders contain identical information at this stage.
3. In the new mobile site folder, add the mobile theme to the folder `themes`. You may also delete any graphical themes that are not required by the mobile version of your site.
4. In the mobile site folder, edit the file `settings.php` and look for the section labeled "Variable overrides." Update the default theme to "mobile" and uncomment the relevant lines. Before editing, the code will appear as follows:

```
# $conf = array(
#   'site_name' => 'My Drupal site',
#   'theme_default' => 'minnelli',
#   'anonymous' => 'Visitor',

... approximately 50 lines
# );
```

After editing, it will appear as follows (note the **bold** lines have changed):

```
$conf = array(
  #   'site_name' => 'My Drupal site',
'theme_default' => 'mobile',
  #   'anonymous' => 'Visitor',

... approximately 50 lines
);
```

Your new mobile site is now ready for use! It uses the same database as the main site and, therefore, will always be exactly in sync with the main site. No extra work is required on your part!

Summary

This chapter addressed ways to modify the preprocess function so that you can prepare and alter page template variables, and alert Drupal of new page templates. More specifically, you learned how to perform the follow tasks:

- Dissect a theme into its component template files
- Use sitewide variables in page templates
- Create new sitewide variables with preprocess functions
- Establish a grid for a page template through custom regions
- Configure a sitewide search block
- Change page templates based on taxonomy, page alias, and content type
- Create and implement print-friendly pages using CSS and the Print module
- Create a low-bandwidth site for mobile devices

In the next chapter you will learn how to fill up the "content" region of your page with nodes that are themed exactly as you want them to be.

Index

E

F

G

H

I

J

L

M

R

S

T

U

V

W

X

Y

Z