

ADOBE DREAMWEAVER CS5 with PHP

from the **SOURCE**

David Powers



Adobe® Dreamweaver® CS5 with PHP: Training from the Source

David Powers

A

Adobe Adobe Press books are published by: Peachpit 1249 Eighth Street Berkeley, CA 94710 510/524-2178 800/283-9444

For the latest on Adobe Press books, go to www.adobepress.com To report errors, please send a note to errata@peachpit.com Peachpit is a division of Pearson Education. Copyright © 2011 David Powers

Acquisitions Editor: Victor Gavenda Project Editor: Rebecca Freed Development Editor and Copyeditor: Anne Marie Walker Production Editor: Becky Winter Technical Editor: Tom Muck Compositor: Danielle Foster Indexer: Rebecca Plunkett Cover Design: Charlene Charles-Will

Notice of Rights

All rights reserved. No part of this book may be reproduced or transmitted in any form by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. For information on getting permission for reprints and excerpts, contact permissions@peachpit.com.

Notice of Liability

The information in this book is distributed on an "As Is" basis, without warranty. While every precaution has been taken in the preparation of the book, neither the author nor Peachpit shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the instructions contained in this book or by the computer software and hardware products described in it.

Trademarks

Adobe, the Adobe logo, and Dreamweaver are registered trademarks of Adobe Systems in the United States and/or other countries.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and Peachpit was aware of the trademark claim, the designations appear as requested by the owner of the trademark. All other product names and services identified throughout the book are used in an editorial fashion only and for the benefit of such companies with no intention of infringement of the trademark. No such use, or the use of any trade name, is intended to convey endorsement or other affiliation with this book.

ISBN-13: 978-0-321-71984-3 ISBN-10: 0-321-71984-0 9 8 7 6 5 4 3 2 1 Printed and bound in the United States of America

Contents

	Introduction
LESSON 1	Why PHP and Why Dreamweaver CS5? 3
	A Rich Mix of PHP Features
	What Is PHP? What Does It Do?
	A Tour of the Main PHP Features in Dreamweaver CS5 9
LESSON 2	Getting Ready to Develop with PHP
	Setting Up a Local Testing Environment
	Checking Your PHP Installation
	Using Virtual Hosts
	Setting Up a PHP Site in Dreamweaver CS5
LESSON 3	A Quick Crash Course in PHP
	How PHP Makes Pages Dynamic
	Taming the Unknown with Variables
	Grouping Related Values in Arrays
	Using Conditions to Make Decisions
	Using Functions to Perform Tasks
	Using Objects and Resources
	Using Operators for Calculations and Joining Strings
	Automating Repetitive Tasks
	Including External Files
	Understanding Error Messages
LESSON 4	Restyling a WordPress Site
	Understanding the Structure of a CMS
	Installing WordPress
	Creating a WordPress Theme

vi	Contents

LESSON 5	Designing and Building Your Own Database	141
	Working with MySQL	.142
	Creating a Database and Tables	.155
	Creating MySQL User Accounts	.161
	Importing Existing Data	165
LESSON 6	Generating PHP Automatically with Server Behaviors	170
	What Server Behaviors Do	.171
	Connecting to the Database.	.171
	Inserting Records into a Table.	.177
	Creating a Login System	.183
	Displaying, Updating, and Deleting Records	.192
	Evaluating the Server Behaviors	207
LESSON 7	Validating Input on the Server	212
	Introducing the Zend Framework	213
	Improving the Registration Form.	.218
	Authenticating User Credentials with Zend_Auth	247
LESSON 8	Zending Email	258
	How PHP Handles Email	.259
	Stopping Spam with a CAPTCHA	.263
	Processing User Feedback	.266
	Processing Other Form Elements.	.277
	Resetting Forgotten Passwords	.286
	Unsubscribing Registered Users	299
LESSON 9	Uploading Images and Other Files	304
	Understanding How PHP Uploads Files.	305
	Creating an Upload Form.	.306
	Using Zend_File for Uploads.	.308
	Sending Email Attachments	334
LESSON 10	Inserting Data into Multiple Tables	344
	Assessing the Task	345
	Creating the Database Structure	.346
	Building the CMS	349

LESSON 11	Updating and Deleting Files in Related Tables
	Selecting Records with SQL
	Completing the CMS
LESSON 12	Using Ajax to Refresh Content
	Enhancing Pages with Ajax
	Introducing Adobe Widget Browser
	Configuring a Widget
	Creating a Master/Detail Set
	Refreshing a Page Without Reloading
	Creating Clean URLs
LESSON 13	Deploying Your Site Online
	Transferring a Database
	Preparing Your PHP Files
	Setting Up Your Remote Server in Dreamweaver
	Index

Introduction

My first encounter with PHP came about 10 years ago. By that time, I already had plenty of experience developing websites. I had started out writing HTML in a text editor before settling on Dreamweaver as my favorite authoring tool. A new project involved publishing more than 30 articles a day. It was a subscription service, so the site needed to be password-protected and searchable. An ordinary website wouldn't do. That's when PHP came to the rescue.

PHP makes communication with a database a breeze, so content can be stored in the database, making it searchable. Instead of creating a new page for every article, pages are populated dynamically with the requested items. You can also password-protect the administrative or members-only area of a site. PHP does a lot more: It can send email, upload files, and attach files to emails—all of which you'll learn how to do in this book. PHP is also the driving force behind the three most popular content management systems: Drupal, Joomla!, and WordPress.

So, where does Dreamweaver come into the picture? Dreamweaver has supported PHP to some degree since 2002, mainly through server behaviors, which automatically generate PHP code for some basic tasks. But the level of support has taken a quantum leap forward in Dreamweaver CS5. The server behaviors are still there (see Lesson 6), but they take a back seat.

The big changes lie in code hinting, embedded PHP documentation (including examples), autocompletion of variables, automatic discovery of dynamically related files, and—per-haps best of all—the ability to view and navigate through PHP pages without leaving the Document window. As a result, it's now possible to style WordPress, Joomla!, and Drupal in Dreamweaver CS5 without the need to generate static pages. These changes are described in detail in Lesson 1, but in a nutshell they should appeal to designers and developers alike.

PHP's popularity springs from being easy to learn. You can achieve practical results very quickly. Of course, like any skill, becoming an expert takes time and practice. The new PHP features in Dreamweaver CS5 not only help the learning process, but you'll find them even more useful as you gain experience. Dreamweaver is my preferred choice for designing the look of a website and organizing files, but I was beginning to use dedicated PHP authoring tools for the dynamic aspects of development. Dreamweaver CS5 has changed all that. I now have the best of both worlds in the same program.

Who This Book Is for

This is a "beyond the basics" book, so you should already have a solid understanding of how a website is built. You should also have a good understanding of HTML, because PHP code needs to be embedded in the underlying structure of a page to display the dynamic output. It's not necessary to know every tag and attribute, but if you don't know the difference between a and an tag, you'll be lost. All the example files and exercises are styled with CSS, but design is not the focus of this book. You don't need to understand CSS to work through the lessons, but your web development skills would certainly be the better for it. You'll also find it makes it easier to follow Lesson 4, where you create a new WordPress theme.

You don't need prior knowledge of PHP. This book doesn't teach PHP in a formal manner, but Lesson 3 provides a crash course in how to write PHP, and Lesson 5 teaches the basics of database design using MySQL, the most popular open source database.

If you already know some PHP, all the better. This book moves at a fairly rapid pace. Lessons 7–12 make extensive use of the Zend Framework, a powerful library of PHP components that take a lot of hard work out of creating dynamic sites. Lesson 12 also uses the jQuery JavaScript framework. Again, you don't need prior knowledge of jQuery or JavaScript, but it will certainly help.

How to Use This Book

Time is precious, so you probably want to jump straight to the solution for your current problem. If you have considerable PHP experience, that approach might work. However, the majority of readers should start with Lesson 1 and work through each one in sequence because each lesson builds on the previous one. If you skip ahead, you're likely to miss a vital explanation and will need to backtrack anyway.

The "Approximate Time" at the beginning of each lesson is simply an estimate of the time it will take to work through the exercises. Don't regard it as a challenge, and don't feel downcast if you take much longer. Each lesson is packed with information. Take time to absorb it, and break the lesson into smaller chunks to match your own pace.

Most lessons contain reference sections followed by hands-on exercises. Each step explains not only what to do, but also why you're doing it. The idea is to help you think about how you could apply the same techniques to your own projects. This isn't a point-and-click book, but instead is one that aims to stimulate your problem-solving abilities. The more you think, the more you're likely to get out of it.

Accompanying files

The accompanying CD contains all the files necessary to complete the exercises in this book. The only exceptions are the PHP/MySQL development environments described in Lesson 2 and the LightBox Gallery Widget in Lesson 12. PHP and MySQL are updated frequently, so it makes more sense to get the most recent versions from the source. In the case of the LightBox Gallery Widget, one object of the exercise is to show you how to install the Adobe Widget Browser and download widgets from the Adobe Exchange.

Lesson 2 describes how to set up the Dreamweaver site to work through the exercises in this book. The files for each lesson are in folders named lesson01, lesson02, and so on. There are no files for Lesson 13. For each lesson that contains exercises, there are normally three subfolders: completed, start, and workfiles. The workfiles folder is deliberately left empty; it's where you should create and save the files for the lesson's exercises. If you follow this structure, the exercise files will use the common style sheets that are stored in the styles folder.

To save time, many exercises have partially completed pages, which you should copy from the start folder to the workfiles folder for that lesson. The completed folder contains copies of the exercise files shown at various stages of completion.

In Lessons 10 and 11, you should create a folder called cms in the site root. The cms_complete folder contains a full working copy of the completed project.

* NOTE: The files were created on a Windows computer but are fully compatible with Mac OS X. However, the path in library.php needs to be adjusted to match the location of the Zend Framework files. See Lesson 7 for details.

Windows/Mac differences

The few Dreamweaver CS5 and PHP differences between Windows and Mac OS X have been pointed out at relevant places in the book.

Keyboard shortcuts are given in the order Windows/Mac, but in the rare cases where there is no Mac equivalent, this has been pointed out. On some Mac keyboards, the Opt(ion) key is labeled Alt. On a UK Mac keyboard, use Alt+3 to type the hash symbol (#).

Using a multi-button mouse with a Mac is now so common that the instructions refer only to right-click. If you prefer a single-button mouse, use Ctrl-click.

Code portability

One of the pleasures of working with PHP is that it's platform-neutral. All the PHP code in this book works equally well on Windows, Mac OS X, and Linux. However, it's important to realize that different versions of PHP and MySQL have different functionality. Also, server administrators have the ability to turn off certain features. To use this book, your web server *must* be running PHP 5.2 and MySQL 4.1 or later. The code will not work with earlier versions.

Getting help

When you encounter a problem, the first person to look to for help is you. Did you skip a step or mistype the name of a variable or function? One of the quickest ways of finding an error is to use Dreamweaver's File Compare feature (choose Help > Using Dreamweaver CS5 > Creating and Managing Files > Comparing files for differences) to compare your file with the version in the completed folder.

File Compare requires a third-party file comparison utility. If you don't have one installed, WinMerge (http://winmerge.org) for Windows and TextWrangler (www.barebones.com/ products/textwrangler/) for Mac OS X are both free.

If you can't solve the problem on your own and a quick search on the Internet doesn't produce the answer, post a question in the Adobe forums. The best one for PHP questions is the Dreamweaver Application Development forum at http://forums.adobe.com/community/ dreamweaver/dreamweaver_development. I'm frequently there providing help, so you might even get an answer from me.

I also post updates and tutorials on my website at http://foundationphp.com/, and you can follow me on Twitter @foundationphp.

Every care has been taken to eliminate errors, but if you think you have found one, please email errata@peachpit.com with the details.

Layout conventions

The following text conventions are used throughout this book:

• **Boldface text.** Words in **bold text** indicate input that you should type in a field or the name of a file you should create.

• Boldface code. Code that is added or changes is displayed in boldface.

```
if ($_POST) {
    if (empty($_POST['username']) || empty($_POST['password'])) {
        $failed = TRUE;
    } else {
        require_once('library.php');
    }
}
```

• Long code. Sometimes, code won't fit on a single line on the printed page. Where this happens, an arrow indicates the continuation of a broken line like this:

• Italics. Text in *italics* is for emphasis or to introduce important concepts.

Let the Journey Begin

Above all, enjoy the experience that lies ahead. Even if you find working with code uncomfortable to begin with, PHP is not hard. Welcome to the ever-growing PHP community. This page intentionally left blank

What You Will Learn

In this lesson, you will:

- Examine the basic structure of Drupal, Joomla!, and WordPress
- Install WordPress 3.0 in your local testing environment
- Create a child theme based on the default WordPress Twenty Ten theme
- Use Live View, the CSS Styles panel, and Code Navigator to style WordPress
- Enable site-specific code hints for WordPress
- Edit a WordPress template

Approximate Time

This lesson takes approximately 2 hours 30 minutes to complete.

Lesson Files

Media Files:

lesson04/start/images/birds_bg_gradient.jpg lesson04/start/images/cormorants.jpg lesson04/start/images/cormorants-thumbnail.jpg lesson04/start/images/screenshot.png lesson04/start/images/seagulls.jpg lesson04/start/images/seagulls-thumbnail.jpg

Starting Files:

lesson04/start/auth_keys.txt lesson04/start/wordpress-3.0.zip

Completed Files:

lesson04/completed/functions.php lesson04/completed/header.php lesson04/completed/style.css



LESSON 4 Restyling a WordPress Site

Open source content management systems (CMSs), such as Drupal, Joomla!, and WordPress, take much of the hard work out of creating a dynamic website. WordPress claims—with some justification—that it takes only five minutes to install. The difficult part is trying to style a CMS to give it a unique look. That job is now considerably easier thanks to several new features in Dreamweaver CS5: navigable Live View, dynamically related files, CSS Inspect, and site-specific code hints.

Instead of constantly reloading the site in a browser to see the effect of your changes, you can now redesign your CMS entirely in the Document window. In this lesson, you'll adapt the default theme for a WordPress 3.0 site. The same basic principles apply to styling Drupal and Joomla!

P.M. V	Biras of a Feather
me About Search Categories • Dreamweaver	Dreamweaver CS5 loves WordPress Posted on May 30, 2010 by admin Yes, it's true. You can now edit and style a WordPress site right inside the Dreamweaver
Recent Posts Dreamweaver CS5 loves WordPress	Document window thanks to improvements in Live View, Dynamically Related Files, and Site-Specific Code Hints. And if Drupal or Joomla! is your preferred CMS, no problem Dreamweaver CS5 supports them, too.
May 2010	Using Live View in combination with CSS Inspect and the CSS Styles panel, you can locat the relevant style rules quickly, and see how your changes affect the look of the site without the constant need to save and preview in a browser.

The redesigned WordPress site.

Understanding the Structure of a CMS

Before embarking on restyling a WordPress site, it's worth spending a few moments examining how a CMS like WordPress, Drupal, or Joomla! is structured. A bare-bones Drupal installation consists of more than 460 files in 58 folders; WordPress has nearly 800 files in 79 folders; and Joomla! weighs in at a whopping 3,913 files in 711 folders. Unlike a website built with HTML, these files don't contain any of the site's content. In fact, the only page that most users ever see is index.php.

With the exception of images and other media files, all the content is stored in a database. The job of the army of files is to insert, update, and delete content in the database, and to serve visitors to the site with the information they want to see. If you open index.php in any of the CMSs, you see just a handful of PHP commands. There's nothing recognizable as a web page. Each part of the final web page is generated separately. Different scripts handle the page header, menus, main content, and footer.

This mass of files can be intimidating, even if you have a good understanding of PHP. As a result, many designers opt for using third-party themes (or templates, as Joomla! calls them) to improve the look of their CMS. There are plenty of good themes and templates available, and the default Twenty Ten theme in WordPress 3.0 is very attractive. But with the help of Dreamweaver CS5, it's not difficult to do your own customization-providing you have a strong grasp of CSS.

With a CMS, it's important to apply security fixes as soon as they're released, so you need to install your custom files in a place where they won't be overwritten. The location depends on the CMS you're using:

- Drupal. Create two subfolders called modules and themes in sites/all. The themes folder is where you install third-party themes or create your own.
- Joomla! Create a subfolder in the templates folder.
- WordPress. Create a subfolder in wp-content/themes.

Although the instructions in this lesson concentrate on creating a WordPress theme, the same principles of editing the CSS apply to Drupal and Joomla!



TIP: There's a tutorial by David Karlins on modifying Drupal themes with Dreamweaver CS5 at www.peachpit.com/articles/article.aspx?p=1590589.

Installing WordPress

The following instructions assume you have created a PHP local testing environment as described in Lesson 2, and that your web server and MySQL are running.

Setting up a MySQL database and user account

Before you can install WordPress, you need to create a MySQL database and user account. Both subjects are covered in greater detail in Lesson 5, but the following steps guide you through the process of setting up a WordPress database.

- 1 Load phpMyAdmin in your browser, and log in as the root user if necessary.
- **2** In the "MySQL localhost" section in the center of the screen type **wordpress** in the "Create new database" text field. Leave all other settings at their default, and click Create.

MyS	QL localhost		
港	Create new database	0	
	wordpress	Collation	- Create
41 12	MySQL connection collation	on: utf8_general_ci	• @

You should see a message that the database has been created. You don't need to create any tables. WordPress does it for you.

3 Click the Home icon 🚮 at the top left of the phpMyAdmin screen to return to the previous page. Then click the Privileges tab at the top of the screen.

CAUTION! Don't be tempted to click the Privileges tab on the previous screen. You must return to the phpMyAdmin welcome page to access the correct screen.

- 4 Click the "Add a new User" link halfway down the screen.
- 5 In the "Add a new User" section, type wpuser in the "User name" field.
- 6 Select Local from the Host menu to insert localhost in the Host field.
- 7 Type P3@chp!T in the Password field, and again in the Re-type field.

📽 Add a new User				
Login Information				
User name:	Use text field:	٣	wpuser	
Host:	Local	•	localhost	1
Password:	Use text field:	•	•••••	
Re-type:			•••••	
Generate Password:	Generate			

8 Scroll to the bottom of the page, and click Go.

phpMyAdmin reports that it has created the user and displays a page where you can edit the user's privileges. The first section, "Global privileges," gives the user the same privileges on all databases, which is insecure.

9 Scroll down to "Database-specific privileges" and select wordpress from the menu labeled "Add privileges on the following database."



This loads a new screen where you define the database-specific privileges.

- **10** You need to select all checkboxes except the three in the Administration box. The quickest way is to click "Check all," and then deselect the three Administration checkboxes.
- **11** Click the Go button in the "Database-specific privileges" section.

Se Edit Privileges: User 'wpuser'@'localho	sť - Database wordpress
Database-specific privileges (Check All / Uncheck Note: MySQL privilege names are expressed in English Structure SELECT SINSERT UDDATE DELETE SINCK DELETE SINCK DELETE SINCK	All) Administration GRANT CCK TABLES REFERENCES
	Go

V CAUTION! There are two Go buttons on this page. Make sure you click the top one.

You should see a message saying you have updated the privileges for 'wpuser'@'localhost'. You're now ready to install WordPress.

Adding WordPress to the phpcs5 site

Installing WordPress involves unzipping the files into the folder where you want to locate the CMS. This can be the site root or a subfolder. For this lesson, use a subfolder of the phpcs5 site you set up in Lesson 2. After extracting the files, you need to edit a configuration file filling in the details of the MySQL database. The rest of the installation process is automated.

- **1** Use lesson04/start/wordpress-3.0.zip or download the most recent version of WordPress from http://wordpress.org/.
- **2** Extract the files to the phpcs5 site root. This should create a folder called wordpress inside the phpcs5 site. The folder contains about 25 files and three subfolders: wp-admin, wp-content, and wp-includes.
- **3** Click the Refresh icon **C** in the Dreamweaver Files panel to see the newly added folders and files.



4 Double-click wp-config-sample.php in the wordpress folder to open it in the Document window, and switch to Code view.

The first part of the script (around lines 18–34) defines the MySQL settings for the CMS. Replace the placeholder text in the first three lines with the name of the database, the user name, and password that you created in the previous section like this:

```
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');
/** MySQL database username */
define('DB_USER', 'wpuser');
/** MySQL database password */
define('DB_PASSWORD', 'P3@chp!T');
```

5 Scroll down to around line 45 to the following section of code:

```
define('AUTH_KEY',
define('SECURE_AUTH_KEY',
define('LOGGED_IN_KEY',
define('NONCE_KEY',
define('AUTH_SALT',
define('SECURE_AUTH_SALT',
define('LOGGED_IN_SALT',
define('LOGGED_IN_SALT',
define('NONCE_SALT',
define(
```

This defines a series of measures designed to make it extremely difficult, if not impossible, for anyone to reuse cookies if the security of your site is breached. When creating your own WordPress site, you can use your own imagination to create unique character sequences, or you can use the automatic key generator at https://api.wordpress.org/secret-key/1.1/salt/.

For this lesson, use the values in lesson04/start/auth_keys.txt to replace the eight lines shown here.

NOTE: In the event that the security of a live WordPress site is breached, you should replace these eight values and update the file on your remote server immediately.

- 6 Save wp-config-sample.php as wp-config.php, and close both files.
- 7 Launch your browser, and open wordpress/wp-admin/install.php in your phpcs5 site. The URL depends on how you set up your testing environment:
 - Virtual host. http://phpcs5/wordpress/wp-admin/install.php
 - Localhost. http://localhost/phpcs5/wordpress/wp-admin/install.php

- **NOTE:** If you are using the MAMP default ports on a Mac, add :8888 after phpcs5 for a virtual host, or after localhost.
 - 8 The install page asks for some basic information to set up the site. Type **Birds of a Feather** in the Site Title field.
 - **9** Leave username at the default admin.
- 10 Type C0rm0R@nT in both Password fields.
- **11** For a live site, you should use a real email address in Your E-mail, because it's used to send alerts about posts waiting for approval. It's also used if you forget your password. A dummy address is fine for testing.
- **12** Deselect the checkbox that allows your site to appear in search engines like Google and Technorati. You won't be deploying this exercise on the Internet.
- **13** Click Install WordPress. In a few moments, you'll see a screen telling you that WordPress has been installed and inviting you to log in as admin.
- 14 Click Log In to open the login screen. Type admin in the Username field and C0rm0R@nT in the Password field. It's also a good idea to select the Remember Me checkbox to avoid the need to type these details every time.
- **15** Click Log In to enter the WordPress Dashboard, the administration center for a WordPress site.





16 Click the name of the site (Birds of a Feather) next to the WordPress logo at the top of the page to view the front page.

Why Can't I See the Multiscreen Button?

The screen shot on the next page shows a Multiscreen button in the Document toolbar, which isn't part of a default installation of Dreamweaver CS5. It comes from the HTML5 Pack that was released in May 2010, a month after Dreamweaver CS5 became available for purchase. Although the Multiscreen button isn't used in this book, the HTML5 Pack upgrades the version of the WebKit browser engine used in Live View to support CSS3 properties that are used later in this lesson.

If you don't see the Multiscreen button, check the status of the HTML5 Pack at http://labs.adobe.com/technologies/html5pack/. It's possible that during the lifetime of this book the pack's functionality will be added to Dreamweaver through the Adobe Updater. Download and install the HTML5 Pack using whichever method is available.

17 In Dreamweaver, double-click index.php in the wordpress folder to open it in the Document window. In Code view, there are just two lines of PHP code, together with a dozen or so lines of comments.

Switch to Design view, and click the Live View button. After a few moments, you should see the Birds of a Feather site in the Dreamweaver Document window. There are several files called index.php in a WordPress site. If you don't see the front page of the Birds of a Feather site in Live View, make sure you opened index.php in the top wordpress folder.



The default Twenty Ten theme in WordPress 3.0 has been designed to look good straight out of the box. But with the help of Dreamweaver CS5's new features, you'll learn how to develop your own theme to style WordPress.

Creating a WordPress Theme

Developing a WordPress theme from scratch requires considerable knowledge of CSS, HTML, PHP, and the WordPress application programming interface (API). The good news is that you can stand on the shoulders of others to adapt an existing theme as a *child theme*.

Child themes work on a similar principle to CSS. The child theme automatically inherits all the features of the existing theme, but you can decide which elements to override. The advantage is that the original files remain intact, so you can revert to the default if you change your mind or make a mistake. Also, if the parent theme is updated, you can replace all its files without worrying about losing your customizations because they're all stored in the child theme. Most themes can be adapted as child themes. Before doing so, check the license. Some commercial themes have restrictions on how they can be used. The Twenty Ten theme used in the following exercises is released under the GNU General Public License (www.gnu.org/licenses/ gpl.html), which means you are free to modify and redistribute it.

Preparing the files for a child theme

Themes consist of at least one style sheet and a number of WordPress templates. A WordPress template doesn't control a complete page. It's more like a Dreamweaver Library item in that it represents a fragment of a page. Each template is named after the part of the page it controls. For example, the Twenty Ten theme has templates called comments.php, footer.php, header. php, sidebar.php, and so on. If you open any of these files, you'll see a mixture of HTML and PHP. If you don't have any PHP experience, the code probably looks incomprehensible, but much of it is based on conditional statements. You'll gain plenty of experience with conditional statements in later lessons, so the code should be a lot easier to decipher by the time you have completed this book.

However, you don't really need to worry about the PHP code in the templates. A child theme requires only one file—a style sheet, which must be called style.css and reside in the child theme's top level folder. The child theme automatically uses the parent theme's templates and custom functions. In other words, at its simplest level, creating a child theme is just the WordPress way of attaching your own style sheet to an existing theme. But if you're feeling more ambitious, you can create your own templates and functions. When the active theme is a child theme, WordPress always looks first in the child theme's folder. If it finds the appropriate template or function there, it uses it. Otherwise, it uses the version in the parent theme's folder. For example, if you create your own version of header.php, WordPress uses it. But if you don't have your own version of footer.php, WordPress uses the one from the parent theme. This gives you the opportunity to experiment. You can copy a template file from the parent theme, and make some changes. If you like the result, you're on the way to developing your own theme. If it doesn't work, just delete the template file from your child theme's folder, and revert to the parent template.

Developing WordPress themes is a vast subject, so the exercises in this lesson only scratch the surface, but they demonstrate how quickly you can begin to style a WordPress site in Dreamweaver CS5.

 In the Dreamweaver Files panel, expand the wordpress and wp-content folders, select the themes folder, right-click, and choose New Folder. Name the new folder **birds_phpcs5**. The new folder should be inside the themes folder at the same level as twentyten.

😑 🧰 wordpress	Fold
😟 🗀 wp-admin	Fold
😑 🦳 wp-content	Fold
😟 🖂 plugins	Fold
E-C themes	Fold
🗁 birds_phpcs5	Fold
i⊞ Ci twentyten	Fold
index.php	1KB PHP
index.php	1KB PHP
😟 🗀 wp-includes	Fold
index.php	1KB PHP

- 2 Expand the twentyten folder, and double-click style.css to open it in the Document window.
- **3** The first eight lines of style.css look like this:

This tells WordPress what the theme is called, plus some basic information about the theme.

- 4 Choose File > Save As or press Ctrl+Shift+S/Shift+Cmd+S. In the Save As dialog box, navigate to the birds_phpcs5 folder, and save the file with the same name (style.css). When asked if you want to update links, click No.
- 5 Close the original style.css, and make sure you're working in the version in the birds_phpcs5 folder. The file path should be visible in the Browser Navigation toolbar.

style.css ×				
Code Split Design	Live Code	(Live View) Inspect	G. Multisers	E C
🛛 💠 🗘 🏠 Address:	file:///Cl/vhosts/phpcs	5/wordpress/wp-contert/	themes/birds_php	ocs5/style.cs
1 /* 2 Theme Name 3 Theme URI:	: Twenty Ten http://wordpres:	s.org/		

Alternatively, click the Open Documents icon 🕒 at the top of the Coding toolbar to reveal the file path.

6 The Theme Name comment must contain a unique name, which cannot consist only of numbers (that's why it's "Twenty Ten," not "2010"). When creating a child theme, you need to specify the parent theme as the child theme's template. Without these two changes, WordPress won't recognize your child theme. Changes to the remaining comments are optional. Amend the comments at the top of style.css like this:

The parent is identified by Template followed by a colon and its folder name.

- 7 Save style.css and close it.
- 8 This is sufficient for WordPress to identify the new theme, but it's a good idea to add an image to distinguish it from others in your Dashboard. The image should be about 300 pixels wide and must be called screenshot.png. Copy screenshot.png from lesson04/start/ images to the birds_phpcs5 folder. The child theme folder should now contain two files.



You'll add more files later, but that's sufficient for now.

TIP: If you create a mockup of your final design in a graphics program, such as Fireworks or Photoshop, you can create screenshot.png by scaling the mockup and exporting it as a .png file. If it can't find screenshot.png, WordPress displays a text description of the theme.

Activating the child theme

The child theme needs to be activated before you can style it in Dreamweaver.

- **1** In your browser, log into the WordPress Dashboard. Depending on how you set up your testing environment, the URL should be one of the following:
 - Virtual host. http://phpcs5/wordpress/wp-admin/
 - Localhost. http://localhost/phpcs5/wordpress/wp-admin/

2 Expand the Appearance section in the column on the left of the Dashboard, and select Themes. The new Birds of a Feather theme should be displayed in the Available Themes section.



- **3** Click the Activate link for the Birds of a Feather theme. After a few seconds, Birds of a Feather is displayed as the Current Theme and Twenty Ten moves down to the Available Themes section.
- **4** Click the Widgets link in the Appearance menu to open the widget settings screen. The child theme inherits the default settings of the Twenty Ten theme. You can study all the options later. For now, remove the Recent Comments and Meta widgets from the Primary Widget Area by dragging them from the sidebar on the right back to the Available Widgets area.
- **5** Drag the Categories widget to move it just below the Search widget. The Primary Widget Area should now look like this:

he primary widget area	
Search	v
Categories	v
Recent Posts	٧
Archives	v

- 6 Click the Background link in the Appearance menu. This allows you to set a background image and color for the site. However, it does this by generating a <style> block in the head of the page. It's better to use style.css to handle this, so leave this screen unchanged.
- 7 Click the Header link in the Appearance menu. This is one of the cleverest parts of the new Twenty Ten theme. It offers a choice of eight header images for your site. You can also upload your own images. The problem is that if your image isn't exactly the same size as used by Twenty Ten (940 × 198 pixels), you're prompted to crop it. WordPress doesn't crop your original, but instead makes a copy. If your image's height is less than 198 pixels, it's stretched. The result is often unsatisfactory.

The height and width of the header images are controlled by a custom function in the Twenty Ten theme. To change the default values, you need to override that function. Leave your browser open at the current page, and return to Dreamweaver.

8 Custom functions for WordPress themes are stored in a file called functions.php. If you attempt to redefine an existing function, PHP throws a fatal error, but WordPress overcomes this problem with a simple conditional statement. All the functions in the parent theme's functions.php file are wrapped in a conditional statement that checks whether a function of the same name has already been defined. If it hasn't, the parent theme defines the function. Otherwise, it uses the one defined by the child theme.

Choose File > New. Set Page Type to PHP, set Layout to <none>, and click Create. Switch to Code view, and delete all the HTML code inserted by Dreamweaver. You should have a completely blank page.

- **9** Add an opening PHP tag at the top of the new page, and save it as **functions.php** in the birds_phpcs5 folder.
- **10** Double-click functions.php in the twentyten folder to open it in the Document window. The file contains extensive comments that help you understand what the functions are for and how to override them.
- **11** Scroll down to around line 47 to locate the following code:

```
if ( ! isset( $content_width ) )
  $content_width = 640;
```

This defines the width of the main content <div> in the Twenty Ten theme. As you can see, the conditional statement sets the value to 640 (pixels) only if \$content_width hasn't already been defined. So, to change the width to a different value, you need to add \$content_width to functions.php in the child theme. Otherwise, this value is used.

12 The header image for Birds of a Feather is 20 pixels wider than the Twenty Ten images, so you can expand the content by the same amount.

Switch to the empty functions.php file you created for the child theme, and add the following code after the opening PHP tag:

```
<?php
$content_width = 660;
```

13 Switch back to the Twenty Ten version of functions.php, and scroll down to locate the following line of code (around line 53):

if (! function_exists('twentyten_setup')):

This conditional statement checks whether a function called twentyten_setup() has already been defined. If it hasn't, it creates the function, which—as the name suggests defines the default settings for the Twenty Ten theme.



}

* **NOTE:** This conditional statement ends with a colon rather than an opening curly brace. This is an alternative syntax for control structures. See http://docs.php.net/manual/en/ control-structures.alternative-syntax.php.

14 To create your own default settings for the child theme, you need to copy the function definition to the version of functions.php in birds_phpcs5. The function definition begins like this (around line 75):

```
function twentyten_setup() {
```

```
// This theme styles the visual editor with editor-style.css to match
\Rightarrow the theme style.
add_editor_style();
```

The final section of the function definition looks like this (around lines 171–178):

```
'sunset' => array(
    'url' => '%s/images/headers/sunset.jpg',
    'thumbnail_url' => '%s/images/headers/sunset-thumbnail.jpg',
    /* translators: header image description */
    'description' => __( 'Sunset', 'twentyten' )
  )
));
```

Select the entire function description, and copy it to your clipboard.

15 Paste the function definition into functions.php in the birds_phpcs5 folder. If you copied and pasted the code correctly, Dreamweaver should display "No syntax errors" in the Info Bar at the top of the Document window.

```
16 Scroll down to locate this code (around line 35):
```

```
define( 'HEADER_IMAGE', '%s/images/headers/path.jpg' );
```

This defines the default header image for the Twenty Ten theme (the tree-lined path). To display a different image, change the filename like this:

```
define( 'HEADER_IMAGE', '%s/images/headers/cormorants.jpg' );
```

You'll add this and other images to the relevant folder shortly.

17 The next section of code defines the width and height of the header image. Change the width from 940 to **960** and the height from 198 to **150** like this:

```
define( 'HEADER_IMAGE_WIDTH', apply_filters(
    'twentyten_header_image_width', 960 ) );
define( 'HEADER_IMAGE_HEIGHT', apply_filters(
    'twentyten_header_image_height', 150 ) );
```

18 About 20 lines farther down is a long section of code that begins like this:

```
register_default_headers( array(
   'berries' => array(
   'url' => '%s/images/headers/berries.jpg',
   'thumbnail_url' => '%s/images/headers/berries-thumbnail.jpg',
   /* translators: header image description */
   'description' => __( 'Berries', 'twentyten' )
   ),
```

This passes a multidimensional array to register_default_headers(), a function new to WordPress 3.0, which defines the choice of header images offered by the theme. The default Twenty Ten images are all 940 pixels wide and 198 pixels high, so they won't fit the child theme.

The media files for this lesson contain two header images called cormorants.jpg and seagulls.jpg, together with two smaller versions called cormorants-thumbnail.jpg and seagulls-thumbnail.jpg. Change all instances of berries in the multidimensional array to **cormorants**, and cherryblossom(s) to **seagulls**. There are only two header images, so you need to delete the other six subarrays. When you have finished, the final section of functions.php should look like this:

```
register_default_headers( array(
    'cormorants' => array(
    'url' => '%s/images/headers/cormorants.jpg',
```

```
'thumbnail_url' => '%s/images/headers/cormorants-thumbnail.jpg',
    /* translators: header image description */
    'description' => __( 'Cormorants', 'twentyten' )
),
    'seagulls' => array(
    'url' => '%s/images/headers/seagulls.jpg',
    'thumbnail_url' => '%s/images/headers/seagulls-thumbnail.jpg',
    /* translators: header image description */
    'description' => __( 'Seagulls', 'twentyten' )
)
);
```

}

19 Save functions.php and copy cormorants.jpg, cormorants-thumbnail.jpg, seagulls.jpg, and seagulls-thumbnail.jpg from lesson04/start/images to twentyten/images/headers. The images must go in the parent theme's folder because that's where register_default_headers() expects to find them.

20 Return to the Header page in the WordPress administrative area. Click the Background link in the Appearances menu, and then click Header to reflect the changes you have made. The Custom Header section should now display the two Birds of a Feather header images, and the text in the Upload Image section should show the new default dimensions of 960×150 pixels.



If necessary, compare your code with lesson04/completed/functions.php.

Styling the child theme

All that remains now is to adjust the styles to give the theme its own look. Most of the remaining tasks are done in Live View and the CSS Styles panel.

- 1 Close functions.php if it's still open, and create a new folder called images in the birds_phpcs5 folder. Copy birds_bg_gradient.jpg from lesson04/start/images to the new folder. You'll use this later as a background image to the new theme.
- **2** Double-click index.php in the main wordpress folder to open it in the Document window. Switch to Design view if necessary, and click Live View. The Birds of a Feather site should display with the new default header and the edited sidebar.



3 The header image is now wider than the menu bar. To fix that, click the Inspect button in the Document toolbar. As you move your pointer over Live View, you'll see different sections of the page highlighted. The content of an element is light blue or aqua, padding is mauve, and margins are yellow.

Notice that as you move from element to element, the currently highlighted element is also selected in the Tag selector at the bottom of the Document window. When your pointer is over the black menu bar below the header image, you should see <div#access> highlighted in the Tag selector. Click the menu bar to select it.

4 Selecting an element turns off the Inspect button, allowing you to move your pointer without highlighting other elements. Open the CSS Styles panel by clicking its tab or by choosing Window > CSS Styles. On Windows, you can also use the keyboard shortcut Shift+F11 (there is no Mac shortcut).

Make sure the Current button is selected at the top of the CSS Styles panel and that the Rules pane is visible in the middle section. If the middle section is titled About, click the Cascade icon as indicated in the following screen shot. You might need to close other panels and drag the panes inside the CSS Styles panel to see the rules and properties listed.



The Rules pane displays all the style rules that affect the current selection. Sometimes you need to examine several rules before finding the right one, but on this occasion, it should be the one selected by Dreamweaver. It's the #access rule shown in the preceding screen shot.

Select #access in the Rules pane. This reveals that the width property is set to 940px. Click the value to edit it, and change the number to **960**. The px unit is controlled by a separate menu, so you don't need to change it.

background	#000		
display	block		
float	left		
margin	0 auto		
width	960	 px 	
Add Property			

As soon as you press Enter/Return to confirm the edit, the menu bar in Live View expands to match the width of the header image.

5 Although the header image and menu bar are now the same length, there's a gap of about 20 pixels of white space on the left of both elements. Finding the cause of the gap is a process of elimination, but if you look farther down the page, you'll see there's a similar gap on both sides of the horizontal line above the footer.

Click the Inspect button again, position the pointer over the footer so that the full width below the horizontal line is highlighted, and click to select it.

Birds of a Feather Proudly powered

6 In the Rules pane of the CSS Styles panel, #colophon is selected. Examining the Properties pane reveals nothing to help eliminate the gap on the left and right, so start moving up the cascade of rules in the Rules pane. The next property begins with #access.menu-header and has a width property of 940px. Change the number to **960** as you did in step 4.

As soon as you press Enter/Return, the white background expands to create the same gap on both sides of the header image, menu bar, and the horizontal line above the footer. This is progress, but the final design calls for the gap to be eliminated.

7 Click <div.hfeed#wrapper> in the Tag selector at the bottom of the Document window to reveal its properties in the CSS Styles panel. You'll see that the padding property is set to 0 20px. This adds 20 pixels of padding to both sides of the wrapper <div>.

Remove the padding by selecting it in the Properties pane of the CSS Styles panel and clicking the trash can icon at the bottom right of the panel.

The left and right sides of the heading image, menu bar, and horizontal line above the footer are now flush with the white background of the wrapper.

8 There's a large gap between the white background and the top of the page. It's caused by the margin-top property, which is set to 20px. Select margin-top in the Properties pane for #wrapper and click the trash can icon to delete it.

The entire contents of the page move up to eliminate the gap, leaving the white background flush with the top of the Document window.

9 With #wrapper still selected in the Rules pane of the CSS Styles panel, change the background property from #fff (white) to #FAF2EF (light pink).

TIP: Hexadecimal values for colors are case insensitive. The Twenty Ten style sheet uses a mixture of uppercase and lowercase, indicating that it's almost certainly the work of more than one person. Color values can also be shortened to three characters if each even character is the same as the preceding odd one. Thus, #fffffff can be shortened to #fff, but #FAF2EF cannot be shortened.

10 The next step is to change the background of the whole page. Begin by selecting <body.home blog> in the Tag selector. The Rules pane selects the body, input, textarea style rule, which covers too many elements, so start moving up the list of rules. The next one, body, defines the background property, which is the one you need to change.

The background shorthand CSS property is difficult to define directly in the Properties pane of the CSS Styles panel, so select the property and click the Edit Rule icon at the bottom right of the panel to open the CSS Rule Definition dialog box.

Properties for "body"					
background					
Add Propert	X.				
i≣ Az↓**↓					
DATABASES	BINDINGS SERVER BEHAVIORS	Edit Rule			

11 The CSS Rule Definition dialog box should automatically select the Background category.

Change the value of Background-color from #f1f1f1 to #E1DFE0.

Click the Browse button next to the Background-image text box, navigate to the birds_phpcs5/images folder, and select birds_bg_gradient.jpg.

Set the Background-repeat menu to repeat-x.

When you click OK, the page background should change from light oatmeal to a vertical gradient that fades from light purple to a light gray.

12 Now comes a little bit of CSS magic—swapping the sidebar from right to left. The default style rule for the sidebar floats it to the right in a margin created by the main content. You can move the sidebar to the other side of the page by floating it left and giving it a large enough negative left margin to sit on the opposite side of the main content. But first, you need to adjust the margins of the main content.

Click the Inspect button and select the main content on the left of the page. The style rule that controls its margins is applied to the container <div>, so click <div#container> in the Tag selector at the bottom of the Document window. The Properties pane of the CSS Styles panel reveals that its margin property is set to 0 -240px 0 0. In other words,

a space of 240 pixels has been created on the right for the sidebar. You need to move the space to the opposite side. Change the margin property to **0 0 240px**. This moves the main content to the right of the page, but pushes the sidebar below it.

TIP: The Twenty Ten style sheet's use of a negative right margin on the container <div> is rather unconventional. It's needed because the width property of the <div> is set to 100%. Using a negative value reduces the width and makes room for the sidebar. If you need to brush up on your knowledge of CSS, take a look at my book, Getting StartED with CSS (friends of ED, 2009, ISBN: 978-1-4302-2543-0).

13 Click the Inspect button again, and move your pointer until the whole of the sidebar is highlighted. Then select it. This selects <ul.xoxo> in the Tag selector. The rule that you want to edit is the next one up the page hierarchy. Click <div.widget-area#primary> in the Tag selector.

The Properties pane of the CSS Styles panel displays no styles, so move up the cascade in the Rules pane. The next one—#primary, #secondary—displays the rules that you need to change.



- **14** Change the value of float from right to left. The sidebar jumps to the left of the page but still below the main content.
- **15** Click Add Property, type **margin-left**, and set the value to **-1180px**. (That's *minus* 1180 pixels.) The sidebar is after the main content in the underlying HTML markup, but a combination of the left float and the large negative margin allows it to leapfrog over the main content and move into the correct position in the margin on the left.
- **16** Let's add a touch of CSS3 coolness to the main content. Click the Inspect button, move the pointer over the main content until it's highlighted, and click to select it. The Tag selector shows you have selected <div.post-1 post type-post hentry category-uncategorized#post-1>. Whew! That's a complex CSS selector. Fortunately, the Rules pane selects the lowest part of the cascade, the style rule for the hentry class, which currently sets only the margin property.

Although you could add the next set of CSS properties through the CSS Styles panel, it's a lot easier to work directly in the style sheet. The problem is that the style sheet contains more than 1,000 lines. A quick way to locate the correct rule is to use the Code Navigator.

Hold down the Alt key on Windows or Cmd+Opt on a Mac, and click anywhere in the "Welcome to WordPress" default post to invoke the Code Navigator, a context-sensitive tool for investigating styles that affect the area you clicked. Many rules affect this area, but you should find .hentry listed near the bottom of the panel, as shown in the following screen shot.

Hello world!

TIONO		
Posted	index.php	-
	Source Code (turns off Live Code)	
Welco	style.css	
	html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6, p, blockquote, pre, a, abbr, acronym, address, big, cite, code, c	lel,
Posted i	dfn, em, font, img, ins, kbd, q, s, samp, small, strike, strong, sub, sup, tt, var, b, u, i, center, dl, dt, dd, ol, ul, li, fieldset, form, la	bel,
- B	legend, table, caption, tbody, tfoot, thead, tr, th, td	
	body	
	body, input, textarea, .page-title span, .pingback a.url	
	body	
	body, input, textarea	
	#access .menu-header, div.menu, #colophon, #branding, #main, #wrapper	•
	#wrapper	
	#main	
	#container	
	#content	
nt> <div.po< th=""><th>#content</th><th></th></div.po<>	#content	
	#content, #content input, #content textarea	
B 2 3		
	.entry margin: 0 0 48px 0; V	
	p	
roperties	#content p, #content ul, #content ol, #content dd, #content pre, #content hr	Ŧ
	Alt+click to show 🗹 Disable 🕸 ind	cator

Click the .hentry link in the Code Navigator to open the style sheet in Split view. Using this technique positions the insertion point directly inside the .hentry style rule.

17 Amend the style rule like this:

```
.hentry {
  margin: 0 0 48px 0;
  padding:5px 5px 5px 15px;
  background-color: #FFF;
  -moz-border-radius: 20px;
  -webkit-border-radius: 20px;
  border-radius: 20px;
  -moz-box-shadow: 10px 10px 5px #888;
  -webkit-box-shadow:10px 10px 5px #888;
  box-shadow: 10px 10px 5px #888;
  border: 1px solid #eee;
}
```

The properties beginning with -moz and -webkit are browser-specific implementations of the CSS3 border-radius and box-shadow properties. Putting the properties in this order ensures that the effects will be maintained when the official properties are implemented by browsers.

If you refresh Live View by pressing F5 or clicking anywhere in Live View, you'll see the post now has a white background with rounded corners. In the Mac version of Dreamweaver, you'll also see a drop shadow as shown in the following screen shot.



NOTE: The Windows version of Live View fails to render the drop shadow correctly. However, if you view the finished WordPress site in a recent version of Safari, Google Chrome, or Firefox, the drop shadow is rendered correctly on both Windows and Mac OS X. Internet Explorer 8 and earlier ignores the CSS3 properties and just displays a rectangular white background.

18 Save style.css.

Hopefully, by now you've got the picture. You use the Inspect button or Code Navigator in Live View to identify elements on the page and inspect the style rules that govern their display. The Tag selector at the bottom of the Document window shows you where the element resides in the document hierarchy. Most CSS properties are inherited, so you often need to go back up the hierarchy to find the element where a specific rule has been applied. The CSS Styles panel in Current mode also allows you to work back up the cascade of rules affecting the selected element.

Styling a web page requires patience and skill. If you have a good command of CSS, you'll find working in Live View with these tools make styling WordPress, Joomla!, or Drupal very similar to working with a static HTML page. This lesson has concentrated on styling the front page of a WordPress site, but Live View is navigable in Dreamweaver CS5. Just hold down Ctrl/Cmd while clicking a link, and you can inspect and style all pages and views within a CMS.

To round out this lesson, let's take a look at a WordPress template and make a slight change so that the text heading can be hidden from visual browsers. When editing CMS templates it's a good idea to set up site-specific code hints.

Enabling site-specific code hints for WordPress, Drupal, and Joomla!

WordPress, Drupal, and Joomla! use many custom functions to generate the content for each page. For example, WordPress provides the bloginfo() function (http://codex.wordpress.org/Template_ Tags/bloginfo) to display information about your site. Dreamweaver uses code introspection to generate hints for your chosen CMS, speeding up editing page templates or functions.

The following instructions show how to enable site-specific code hints for WordPress in the phpcs5 site. The procedure is identical for Drupal and Joomla!

- 1 Make sure that the phpcs5 site is selected in the Files panel, and that the active document is from the same site or that all documents are closed.
- 2 Choose Site > Site-Specific Code Hints. As long as you haven't previously set up sitespecific code hints for the same site, the Site-Specific Code Hints dialog box automatically recognizes not only which CMS is installed, but also the correct folder.

Site-Specific Code	Hints				×
Structure:	Wordpress	•		+ 📼	0 🗑
Sub-root:	C:\vhosts\phpcs5\wordpress\				6
File(s):	+-				
	Files	Scan	Recursive	Extensions	*
	B Sub-root				
	wp-content	1	1	.nho	=
	themes	1	1	.php	
	mp-admin	1	1	.php	
	wp-includes	1	1	.php	-
	۰ III ۲				,
Update:	C:\vhosts\phpcs5\wordpress				8
	Scan this folder Recursive Exte	insions			
Help			OK		Cancel

- **3** All that's necessary is to check that the Sub-root text field points to the folder that contains the top level of the CMS. If it doesn't, click the folder icon next to the Sub-root text field, and select the correct folder.
- **4** After checking the Sub-root text field, just click OK. That's all there is to it.
- **5** Dreamweaver inserts a file called dw_php_codehinting.config into the site root and automatically cloaks it to prevent it from being uploaded to your remote server when you use site synchronization to update your files. This file is used only in your local development environment.

Editing a WordPress page template

The first time you dive into a WordPress template can be a baffling experience, but the Twenty Ten theme is well commented. If you understand the HTML structure of the template you're working with, it's not too difficult to work out where to add custom features, such as a static paragraph, or remove elements that you don't want. As long as you make the changes to a template in a child theme, you can always delete the template and use the parent theme's default version if you make a mistake.

- **1** Open index.php in the top-level wordpress folder and click Live View to display the front page of the Birds of a Feather site.
- **2** Click Live Code to display in Split view the underlying HTML output generated by WordPress. Adjust Split view so you can see the code and "Just another WordPress site" at the top right of the page (you might need to scroll horizontally in Live View).
- **3** Click between the words "Just" and "another" in Live View. Depending on the size of your monitor, this should scroll Code view so that the equivalent HTML output is in the center with the insertion point between the two words, as shown on line 24 in the following screen shot:



You can't edit the output in Live Code, because it's dynamically generated by WordPress, but inspecting the output here gives you a good idea of what to look for when you open the WordPress template.

The text you clicked is in a <div> that has the ID site-description. You can also see on line 19 that the main heading has the ID site-title, and that the text is wrapped in a link that returns to the front page of the WordPress section of the site.

The final point to notice is that the alt attribute of the header image on line 26 is empty. The HTML specification requires all images to have alternative text (in the alt attribute), but it's recommended to use an empty string when the image is purely decorative. This redesign incorporates text into the image, so the same text should be inserted into the alt attribute in case the image is not displayed for any reason. **4** In the Files panel, double-click twentyten/header.php to open it in the Document window. Save the file as **header.php** in the birds_phpcs5 folder, and click No when asked if you want to update the links. Close the original version of header.php. You want to edit the copy in the child theme.

```
5 Scroll down to locate the following (it should be around line 54):
<div id="site-description"><?php bloginfo( 'description' ); ?></div>
```

This uses the WordPress bloginfo() function to display the site description ("Just another WordPress site").

- **6** Delete the entire line, and save header.php.
- 7 Switch back to index.php, and press F5 or click the Refresh icon *c* in the Document toolbar (not the Files panel) to update Live View. After a few moments, the text disappears from the top right of the page, and the corresponding code is removed from Live Code.
- **8** Switch back to header.php, and examine the code immediately above the line that you removed in step 6. It looks like this:

This code checks whether the current page is the site root or the front page of the WordPress section. If it is one of these, \$heading_tag creates a pair of <h1> tags. Otherwise, it creates a <div>. It then creates a link to the front page of the WordPress site, which is wrapped around the site name.

9 The site name is already in the header image, so you don't want it displayed twice. However, you should leave it in the underlying code for search engines and screen readers for the blind. Edit the code like this to remove the link and tags:

```
<?php $heading_tag = ( is_home() || is_front_page() ) ? 'h1' : 'div'; ?>
  <<?php echo $heading_tag; ?> id="site-title">
      <?php bloginfo( 'name' ); ?>
</<?php echo $heading_tag; ?>>
```

10 Save header.php, and refresh Live View in index.php. The text heading is still there, but it's no longer a link.

11 Click the Inspect button, select the Birds of a Feather text in Live View, and open the CSS Styles panel in Current mode. In the Rules pane, #site-title should be automatically selected. You know this is the style rule you want to change, because it's the ID selector for the element you just edited in header.php.

h1, h2, h3, h4, h3#comments-t	h5, h6 ⊲h1> ¤de, h3 ⊲h1>	E
#site-title	<h1></h1>	-
Properties for "#si	te-title"	
float	left	
font-size	30рх	
line-height	36px	
margin	0 0 18px 0	
width	700px	
Add Property		
t≣≞ Az∔(**∔		● Đ / O 🖯

12 Select each of the properties in turn and click the trash can icon at the bottom right of the CSS Styles panel to delete them. Then click Add Property to add the position property, and set it to absolute.

Click Add Property again to add the top property, and set it to -1000px (minus 1000). The heading disappears from visual browsers but remains accessible to search engines and screen readers.

13 There's still a large gap at the top of the page, so click the Inspect button again, and select the area above the header image.

The #header rule is selected in the CSS Styles panel. Delete the padding property to remove the gap.

14 Click the header image in Live View to select the #branding img style rule, and delete the border-top property to remove the thick black border at the top of the page.

Save style.css to preserve the changes.

15 Return to header.php, and locate the following code, which inserts the header image (around line 63):

```
<img src="<?php header_image(); ?>" width="<?php echo HEADER_IMAGE_WIDTH;

> ?>" height="<?php echo HEADER_IMAGE_HEIGHT; ?>" alt="" />
```

You could hard code the value of the alt attribute, but that would mean changing the template if you decide to use the same theme for different sites. It makes more sense to use the WordPress API.

Position the insertion point between the quotation marks of the alt attribute, type an opening PHP tag followed by a space, and then press Ctrl+spacebar to bring up code hints. Type **bl**. Dreamweaver should select bloginfo().



Press Enter/Return to autocomplete the function name. Dreamweaver automatically inserts the opening parenthesis. Complete the code by typing '**name**' followed by a closing parenthesis, semicolon, and closing PHP tag. The alt attribute should look like this: alt="<?php bloginfo('name'); ?>"

16 Save header.php, and refresh Live View in index.php. When Live Code reloads, you should see "Birds of a Feather" in the alt attribute.

You can check your code against style.css and header.php in lesson04/completed.

What You Have Learned

In this lesson, you have:

- Examined the basic structure of Drupal, Joomla!, and WordPress (page 112)
- Installed WordPress 3.0 in your local testing environment (pages 113–119)
- Created a child theme based on the default WordPress Twenty Ten theme (pages 119–127)
- Used Live View, the CSS Styles panel, and Code Navigator to style WordPress (pages 128–134)
- Enabled site-specific code hints for WordPress (page 135)
- Edited a WordPress template (pages 136-139)

Index

Symbols

! operator, 76, 78-79 != operator, 75 !== operator, 75 ""(double guotation marks), 67-68 # (hash sign), 62 % (percent) operator, 91 \$ (dollar sign), 63 identifying PHP variables with, 448 using as variable, 20 using in passwords, 163 \$() in jQuery, 448-449 & (ampersand), 361 && operator, 76 - operator, 91 -> operator, 225 - operator, 91 '(apostrophe) enclosing string in double guotations when using, 236 unwanted backslash with, 237-238 ''(single quotation marks), 67 () (parentheses), 79-80)) (extra closing parentheses), 283 / operator, 91 * operator, 91 /* and */, 62 + operator, 91 ++ operator, 91 . (period), 323 : (colon), 125 ; (semicolon) comments beginning with, 37 ending PHP statements with, 63 removing to enable PHP extensions, 39 ?> (closing tag), 61, 224 <? (opening tag), 61 < operator, 75 <= operator, 75 = operator, 379 => operator, 225 == operator, 75, 328 === operator, 75 > operator, 75 >= operator, 75 / (backslash), 237-238 [] (square brackets), 225 [(opening square bracket), 225 (vertical pipe), 38 || operator, 76, 283

_ (underscore), 16, 63, 225 ' (backticks), 150 ∞ (infinity symbol), 145

A

activating child themes, 122-127 Active Server Pages (ASP), 29 Add Web Site dialog box, 44 addValidator() method, 312 Adobe Dreamweaver. See Dreamweaver Adobe Dreamweaver Developer Toolbox (ADDT), 208 Adobe Widget Browser, 420-437 about, 420 choosing widgets from Adobe Exchange, 422-424 configuring widgets, 424–427 illustrated, 423, 424 installing and launching, 421-422 Ajax about, 7 refreshing content with, 418, 419 aliases, column, 441 anonymous functions, 450, 453 Apache web servers integrating PHP into, 29 location for document root on, 30 permissions for file uploading on, 309 registering virtual hosts in Windows OS, 42-44 verifying support for URL rewriting, 457–458 virtual host setup on, 41 arguments finding function, 82-84 passing, 80, 359, 361 arithmetic operators, 91 arrays, 70–73. See also specific arrays associative, 71, 287 basic, 70 defined, 61, 70 \$_FILES, 305 names of functions and, 359 opening square bracket added to, 225 superglobal, 71–73 using with quotation marks, 68-69 associative arrays defined, 71 updating records using, 287 attached files, 334-340 authentication types of Zend_Mail, 261 unsupported by mail() function, 260

User Authentication server behaviors for, 183, 185–190 using Zend_Auth component, 247–255 autoloading ZF class files, 221–223 automating tasks with loops, 96–100

B

backgrounds, 131 backing up database with .sql file, 464, 465 php.ini, 37 site definitions, 56 balancing braces in PHP, 12 banner images, 103 BBEdit, 40, 46 Bindings panel, 194–196, 201 BLOB data type, 152, 153 breaking out of loops, 100 browsers previewing upload form design for, 307-308 requesting web page using PHP, 6 building CMSs, 349-375 adding image details in table with script, 358-363 adding photos separately, 371-374 converting Zend_Paginator into server behavior, 390-392 creating management page for places table, 385-390 deleting table records, 405-408 illustrated, 378 inserting data in tables, 368-371 managing and displaying resized images, 392–395 overview of updating and deleting records, 385 photos table updates, 409-413 preparing administration pages for, 349–354 removing photos from CMSs, 414 selecting records with SQL, 379-383 updating records in places table, 396-405 using state_id as foreign key, 354–358 validating and uploading photos, 363-368

С

calculations having databases do, 149 operators in, 91–92 reassigning results to same variables, 95–96 strings in, 93–94 CAPTCHA, 263–265 about, 263 choosing class for, 263–264 using reCAPTCHA, 264, 265 Cartesian joins, 381–382 case sensitivity Apache directive, 41 deselecting for PHP links, 55

keywords and, 65 MySQL and, 150 PHP, 38 PHP scripts and, 106 variables and, 64 chaining Zend_Validate methods, 230, 234-235 CHAR data type, 151–152 characters in PHP variables, 63, 64 Check New Username server behavior, 185 checkboxes handling by \$_POST array, 280-282 using, 277, 279, 280 checking out files, 472 child themes activating, 122-127 adjusting function definitions for, 125–126 defined, 119-120 placing images on Dashboard for, 122 preparing to style WordPress, 120–122 styling, 128-134 classes class constructors, 89 code hinting for, 15-19 commonly used validation, 229 defined, 88 file validation, 313 loading class files for Zend Framework, 221–223 typing in name of, 225 clearIdentity() method, 253, 254 cloaking, 467 closing PHP tags, 61, 224 CMSs. See content management systems code. See also code hinting applying/removing comment tags in, 13-14 book's conformity to XHTML standards, 4 checking insertion in Code or Split views, 285 collapsing PHP, 12–13 copying/pasting from Zend_Paginator as server behavior, 390-392 enabling Live Code, 14-15 highlighting between braces, 12 programming techniques for PHP, 61 Split Code view for, 13 using echo in PHP, 387-389 using PHP code blocks, 61–62 code hinting, 15-20 core PHP elements supported, 15–16 custom function and class, 17-18 enabling site-specific WordPress, 135 establishing site-specific Zend Framework, 215-218 improvements to, 16 recognizing classes and objects, 18-19 selecting for Zend_Auth component, 250 site-specific, 18 tips for, 16-17 turning on/off, 19–20 typing shortcuts recognized with, 225

Code Navigator, 133-134 Code view checking code insertion, 285 removing server behaviors in, 182 collation and sorting, 156-157 color, hexadecimal values of, 131 columns adding index to, 219-220 aliases for, 441 defining and saving multiple, 157–159 naming, 149-150 needed in users table, 157 combined assignment operators, 96 comments adding to child theme, 121-122 adding to scripts, 62 applying and removing tags for, 13-14 comparison operators, 75 compressing data for remote transfer, 463, 464 computer requirements for web server, 28 concatenation, 94 conditional operators, 446-447 conditional statements, 74-79 about, 74-75 checking for values with, 445-446 comparisons in, 75-76 defined, 61 ending with colon, 125 logical Not operator, 78-79 testing multiple conditions with, 76 testing radio button groups with if, 282, 283 values considered false, 79 configuring test environments. See also PHP testing environments activating configuration changes, 39 changing php.ini file, 36-39 checking all configuration settings, 36 setting temporary upload directory, 35 understanding configuration page, 32-33 connecting to database about, 171 creating MySQL connection file, 171–173 denying access during, 226 editing connection files, 175-177 location of Connections folder, 173 troubleshooting connections, 174-175 using Zend_Db object, 223-226 viewing connections files in Dreamweaver, 175 Connections folder, 173 constants, 15-16 constructors for classes, 89 content management systems (CMSs). See also WordPress adding photos separately, 371-374 deleting records from tables, 405-408 developing database structure for, 346-348 editing WordPress page template, 136–139

enabling site-specific code hints for, 135 handling updates to photos table, 409–413 illustrated, 344 inserting data in tables, 368-371 inserting image details in table with script, 358–363 installing WordPress, 115–118 planning new website's, 345–346 preparing administration pages for, 349-354 removing images from, 414 selecting records with SQL, 379-383 structure of, 112 styling and redesigning, 111 updating records in places table, 396-405 using state_id as foreign key, 354-358 validating and uploading photos, 363-368 cookies, 188 CRAM-MD5 authentication, 261 creating PHP sites, 47-54 adding subfolder to server root, 48 backing up and restoring site definitions, 56 creating site definition, 48–52 editing site definitions, 54-56 multiple server configuration, 52–53 testing your testing server, 53-54 using virtual host, 47 Crow's Foot notation, 145-146 CSS Inspect mode, 25 CSS Rule Definition dialog box, 131 CSS Styles panel, 129 currency values in MySQL, 151

D

Dashboard (WordPress) illustrated, 117 placing theme image on, 122 data. See also transferring data to remote database; validating input activating database's dynamic, 172 collation and sort order, 156-157 compressing for transfer, 463, 464 documentation of data types, 82 inserting in CMS tables, 368-371 MySQL data types, 150–155 organizing, 143-144, 148-149 PHP data types, 65 database servers, 28 databases, 141-167 adding extra columns to, 220–221 backing up with .sql file, 464, 465 collation and sort order in, 156-157 connecting with Zend_Db object, 223-226 data organization in, 143-144 defining, 155–156 designing, 141, 148-149 developing structure for, 346-348 displaying record results for, 194–196

displaying results of multiple loops through records, 196-197 importing existing data, 165-167 inserting user details in, 246-247 naming tables, columns, and, 149-150 organizing data in, 143-144, 148-149 paging through results with Zend_Paginator, 383-385 PHP-friendly, 8-9 querying in ZF, 244-245 relational, 144, 345-346 See also, transferring data to remote database setting up MySQL database used with WordPress, 113-114 steps for activating dynamic data, 172 storing images in, 153 transactions in, 160 updating records, 192, 200-204 validating data before inserting in, 212 viewing table definition files for, 175 Databases panel, 172 DATE data type, 154 date formatting in MySQL, 442–447 DATE_FORMAT() function, 441 DATETIME data type, 154 DECIMAL data type, 151 decision chains password reset, 294 reset request, 288 Delete Record server behavior, 204-206 deleting database records, 192, 204-206 images using links on CMS page, 395 records from places table, 405–408 root user account in MAMP, 162 deploying sites online, 460-475 adjusting file paths, user accounts, and passwords, 467-468 creating folder for file uploads, 469-470 file transfers outside site root, 475 removing error messages from scripts, 465-467 setting up FTP access to remote server, 470–475 transferring data to remote database, 462-465 uploading Zend Framework to remote server, 468-469 deprecated errors, 108 Design view, 181, 182 designing databases, 141, 148-149 directives case sensitivity of Apache, 41 configuring PHP Core, 33 display_errors, 34,77 magic_quotes-gpc, 35 register_globals, 34 directories uploading files to temporary PHP, 35 virtual IIS, 41

Display Error Message dialog box, 243 display_errors directive, 34 <div> container defining width of, 124 negative right margin on theme, 132 document root, 30, 40 documentation finding function arguments, 82-84 Zend Framework, 215 do...while loops, 97, 196 dragging/dropping files in Files panel, 474 Dreamweaver. See also creating PHP sites code hinting, 15-20 collapsing code, 12–13 configuring dynamically related files, 22–23 defining and testing PHP site in, 47-54 highlighting code between balancing braces, 12 improved PHP features in, 4, 15-25 inspecting and styling pages in CMS from, 134 line numbering in, 10 Live View navigation in, 24–25 locating ZF site-specific code hints in, 215–218 Multiscreen button in HTML5 Pack, 118 PHP connection files used by, 171–173 PHP features in previous versions, 9–15 real-time syntax checking, 21 server behaviors in, 170-171, 207-209 setting preferences in, 10 setting up FTP access to remote server, 470–475 Split Code view, 13 syntax coloring, 10–12 Drupal. See also WordPress enabling site-specific code hints for, 135 redesigning themes in, 111 structure of, 112 Dynamic Data dialog box, 198, 199

Ε

Easyphp activating configuration changes for, 39 integrating PHP into IIS with, 29 locating document root for web server, 30 using subfolder of server root with, 48 echo, 387-389 Edit Coloring Scheme for PHP dialog box, 11 editing CMS page template, 136–139 connection files, 175-177 database-specific privileges, 163-164 Log In User server behavior to encrypt password, 190-192 PHP configuration to repair scripts, 32–33 php.ini in Windows and Mac OS systems, 37 site definitions, 54-56 WordPress page template, 136–139

elements effect of highlighting, 128-139 selecting HTML elements using Tag selector, 196 elseif keyword, 75 email, 256-301 attachments for, 338-340 creating mail connector script, 266 incorporating reCAPTCHA in validation script, 270-272 mail() function in PHP, 259–260 mailing feedback, 273-276 modifying error message displays for, 267-269 preparing to send with Zend_Mail, 261-263 recipient's address in Windows OS, 259 resetting forgotten passwords via, 286-299 sending attached files, 334-340 stopping spam with CAPTCHA, 263-265 troubleshooting, 276 unsubscribing users via, 299-300 Zend_Mail for, 261-263 email header injection, 258, 260 embedding PHP in page, 61-62 empty fields checking for, 329, 330 preventing errors for, 272 encoding specification for Zend_Mail, 262 encrypting passwords with server behaviors, 190-192 ENUM data type, 152–153 error messages about, 108-109 customizing upload, 332-333 generating when validation fails, 228 information revealed in, 34 modifying existing server behaviors for email, 267-269 providing photo uploading, 367 removing from scripts, 465-467 scripting for ZF login, 249-251 storing in \$errors array, 242 turning off, 34, 77 errors. See also error messages error markers for syntax checker, 21 failure to load next page, 184 handling in finished registration form, 242-244 preventing feedback form, 272 redisplaying checkboxes when form submitted with, 281-282 reporting level for, 38 troubleshooting php.ini configuration, 39 uploading oversize files, 314-318 ZF exception handling with try and catch, 226-227 \$errors array, 242 escape characters in strings, 69 Expand icon, 474 Extensible Hypertext Markup Language (XHTML), 4

Extensible Markup Language (XML), 4 external files with PHP, 101–108 using HTML, 101–105 using scripts, 105–108

F

FALSE keyword, 65, 79 false values in PHP, 79 fatal errors, 108 feedback forms adapting for email attachments, 338-340 emailing, 273-276 preventing errors when fields empty, 272 preventing malicious use of, 258 fetchAll() and fetchRow() methods, 290 fields checking for empty, 329, 330 defined, 149 inserting hidden form, 306–307 preventing errors when data missing from, 272 FIGlet text, 263, 264 file extensions building and validating renamed, 321, 324, 325 enabling other PHP, 38-39 renaming files without, 328 files. See also php.ini file; uploading files adding WordPress media, 126-127 adjusting paths in deployed site scripts, 467–468 checking for username duplication, 245–246 configuring MySQL database, 115–116 displaying validation messages for uploaded, 314-318 dynamically related, 22-23 file validation classes, 313 locating custom CMS, 112 moving to and from remote server, 473-474 MySQL connection, 171–173 overwriting during upload, 305, 326–329 protecting with cloaking, 467 removing spaces from name, 318-321 renaming duplicate uploaded, 321-326, 366-367 sending as attachments, 334-340 setting maximum size of, 306, 312 setting upload destination for, 310 transferring outside site root, 475 using external, 101–108 Files panel, 473-474 \$_FILES array, 305 FLOAT data type, 151 folders creating for remote server uploads, 469-470 Repository view for, 474 selecting for ZF site-specific code hints, 215-218 setting destination for uploaded files, 310, 311 setting up Apache vhosts, 42

structure of CMS, 112 testing uploads with local, 308-309 ZF, 214-215 for loops, 98 foreach loops multiple file uploads with, 330-332 using, 98-99 foreign keys constraints for, 160 defined, 145 inserting state_id as, 354-358 linking tables using, 345 organizing table columns for, 147 forms. See also feedback forms; fields; records; registration forms checkboxes in, 277, 279, 280-282 directing user to login, 189–190 emailing feedback, 273-276 empty fields in, 272, 329, 330 hidden fields in, 306-307 inserting records using online, 177-182 loading records into update, 200-204 multiple-choice <select> lists in, 279, 280, 285-286 name and id attributes in, 178, 179 password reset, 297 radio buttons in, 277, 278, 280, 282-284 resetting forgotten passwords for, 286-299 sending by email, 258, 259-263 single-choice <select> menus, 225, 284-285 testing registration, 182-183 testing submission of with \$_POST, 277 upload, 306-308 FTP access in Dreamweaver, 470-475 function keyword, 85 functions, 79-88 accented characters in names of, 64 anonymous, 450, 453 arguments and parameters of, 80 code hinting for, 15-18 creating resource, 90-91 custom, 84-87 defined, 61, 79 defining and naming, 85 function signature, 359, 361 iQuery, 449-450 location of custom WordPress theme, 124 methods and, 88 overriding in Twenty Ten theme, 124-126 passing arguments in, 80, 359, 361 reading PHP documentation for, 82-84 variable scope, 87, 449

G

Generate Behavior dialog box, 241 Get icon, 473 GET method, 73 \$_GET superglobal array, 71–73 getIdentity() method, 254 getimagesize() method, 446

Η

hasIdentity() method, 253 header adjusting image with style sheet, 130 revising for Twenty Ten theme, 127 Twenty Ten theme link to, 124 .hentry style rule, 132-133 hidden form fields, 306-307 hosting companies checking host name for MySQL, 467 limiting MySQL user accounts, 165 HTML (Hypertext Markup Language) code's conformity to XHTML standards, 4 embedding PHP in, 5 sending email in, 274-275 using as external file, 101–105 using closing PHP tags following, 224 HTML5 Pack upgrade, 118 HTTP tunneling, 464

I

icons Expand, 474 Get, 473 id attributes, 178, 179 if statements, 282, 283 IIS (Internet Information Services) all-in-one packages integrating PHP into, 29 configuring virtual hosts, 42 creating temporary upload directory, 35 integrating PHP into, 29 permissions for uploading files in, 309 registering virtual hosts in, 44-45 virtual hosts unsupported on, 41 images. See also photos table; places table adding separately in CMS, 371–374 adding theme, 122 associating with multiple places, 394 deleting places without deleting, 405–408 displaying resized database, 392–395 inserting details in CMS table with script, 358–363 positioning captions below gallery, 448 refreshing without reloading page, 450–456 removing from CMSs, 414 saving uploaded, 368 scripts validating uploaded, adding theme, 317-318 storing in database or on server, 153 uploading and validating for CMSs, 363-368 importing existing data to MySQL, 165–167 infinity symbol, 145

INNER JOIN about, 381, 382 LEFT JOIN vs., 382-383 Insert Record server behavior, 177-183 Insert Tag Accessibility Attributes dialog box, 307 installations Adobe Widget Browser, 421-422 MAMP, 28, 29 software for PHP testing environments, 28 testing PHP, 31-32 WordPress, 115-118 Zend Framework, 214–215 instantiating objects about, 89-90 Zend_Mail_Transport_Smtp object, 266 INT data type, 151 Internet Information Services. See IIS intval() method, 397 is_int function, 397 is_numeric, 397

J

JavaScript. See also jQuery ensuring page function if disabled, 418, 419 using Ajax, 419 variable scope in, 449 joining strings, 94–95 tables, 381–382 Joomla!. See also WordPress enabling site-specific code hints for, 135 redesigning themes in, 111 structure of, 112 jQuery combining with PHP to refresh pages, 450–457 functions and variable scope, 448–450 refreshing page content without reloading, 448–450

Κ

keys. See also foreign keys; primary keys obtaining public and private reCAPTCHA, 265 primary and foreign, 145

L

LEFT JOIN clause, 382–383 LightBox Gallery widget displaying master list of places, 437–440 folders and files for, 429 inserting, 427–430 jQuery and, 448 populating gallery dynamically, 430–437 positioning captions below images, 448 refreshing images without reloading page, 450–456 script refreshing places, 456–457

lightning bolt icon, 198 line numbering, 10 links creating internal website, 40 nonstandard functions for site root, 55 relative, 251 Linux servers removing spaces from filenames for, 318-321 setting remote server permissions for file uploads, 469-470 Live Code, 14-15 Live View editing CMS template in, 136-139 navigating in, 24-25 refreshing, 134 styling child theme from, 128 testing pages in, 107 troubleshooting file uploading in, 312 loading ZF class files, 221–223 Log In User server behavior editing to encrypt passwords, 190–192 error message omitted with, 249 using, 185-187 Log Out User server behavior, 188-190 logical operators, 76 login systems authenticating Zend_Mail login, 261 controlling user access with Zend_Auth, 248–249 creating, 185–187 displaying user details of logged, 254-255 error message display in, 249-251 logging out with Zend_Auth, 253-254 testing Zend_Auth scripts for, 251–252 tracking logged in users, 188 User Authentication server behaviors and, 183, 185 loops, 96-100 breaking out of, 100 defined, 61 displaying query data from multiple, 196–197 do...while, 97, 196 for, 98 foreach, 98-99 types of PHP, 96 while,97

Μ

Mac OS systems activating code hinting, 17 adding virtual hosts manually, 46–47 configuring WordPress files for MAMP default ports, 117 editing php.ini in, 37 enabling other PHP extensions, 38–39 installing MAMP for, 28, 29 manually adding virtual hosts, 46–47

permissions for uploading files in, 309 virtual hosts created in MAMP, 45 magic guotes dealing with unwanted backslashes, 237-238 magic_quotes-gpc directive controls, 35 mail() function about, 259-260 authentication unsupported in, 260 using Zend_Mail as wrapper for, 261–263 MAMP configuring WordPress files for default ports, 117 deleting root user account in, 162 installing, 29 locating document root for web server, 30 virtual hosts created in, 45 Manage Places page, 385-390 Manage Sites dialog box, 56 many-to-many relationship, 146, 147, 148 master/detail sets, 437-441 about, 437 creating detail page, 440-448 date formatting in MySQL, 442 displaying master list of places, 437-440 MAX_FILE_SIZE hidden value, 306, 312 methods. See also specific methods iQuery, 449 object's access to, 88 Zend_Mail, 262 Microsoft Web Platform Installer (WPI), 29 Microsoft Windows. See Windows OS systems MIME type validation, 322-323 Model-View-Controller (MVC) design pattern, 213 MTA (mail transport agent), 259, 261 multiline comments, 14, 62 multiple-choice <select> lists, 279, 280, 285-286 Multiscreen button (Dreamweaver), 118, 119 MVC (Model-View-Controller) design pattern, 213 MyISAM, 160 MySQL about, 8-9 checking host name for, 467 choosing data types for, 150–155 choosing UI for, 142-143 collation and sort order in, 156–157 configuring WordPress database files, 115–116 connection file for, 171-173 creating WordPress database and user account, 113-114 date and time, 154 defining database in, 155–156 defining users table for user registration, 157-159, 161 formatting date in, 442 hosting company limits on user accounts, 165 importing existing data to, 165-167 limitations in MySQL-based registration form, 218-219

naming tables, columns, and databases, 149-150 number data types, 151 pronunciation of, 150 read/write privileges for, 161-164 registration form user messages in, 218 selecting storage engine in, 160 string data types, 151-154 strong passwords for, 163 timestamps in PHP and, 155 troubleshooting connections, 174–175 user accounts for, 161–165 using with testing environments, 27 view-only user accounts for web connections, 161, 165 MySQL Connection dialog box, 172–173 MySQL Workbench, 142

Ν

name attributes, 178, 179 naming child themes, 121 functions, 85 .sql file automatically, 463 tables, columns, and databases, 149–150 variables, 63-64 Navicat, 142, 143, 464-465 navigating places table query results, 385-390 query results with Zend_Paginator, 383-385 using Live View, 24–25 New Server Behavior dialog box, 239 notices, 108 NULL keyword, 65, 79 numbers Dreamweaver line, 151 MySQL data types for, 151

0

object-oriented programming (OOP), 88 objects about, 88 code hinting for, 18–19 instantiating, 89–90, 266 using as variable, 88 Zend_Db, 223-226, 383 Zend_Mail_Transport_Smtp, 266 ON clause, 381-382 one-to-many relationship converting from many-to-many relationship, 148 defined, 145, 147 illustrating, 145, 146 Open Ajax Alliance, 421 opening Code Navigator, 133–134 opening tags in PHP, 61

operators arithmetic, 91 combined assignment, 96 comparison, 75, 79 defined, 61 logical, 76 ORDER BY clauses optional nature of, 379 specifying sort order with, 380–381 overwriting files during upload, 305, 326–329

Ρ

parameters defined, 80 description in documentation, 83 preventing spam, 259, 260 Parameters dialog box, 198 parse errors, 108, 109 passwords adding encryption for Update Record server behavior, 202-204 adjusting in deployed scripts, 467-468 checking user, 185-187 encrypting, 190-192 protecting web pages with, 187-188, 253 resetting forgotten, 286-299 strong MySQL, 163 uploading forms to site areas protected by, 320 validating strong, 235-236 paths adjusting in deployed site scripts, 467-468 changing when autoloading ZF class files, 222 getimagesize() difficulties with site-rootrelative path, 446 relative, 101 permissions setting for remote server file uploads, 469-470 troubleshooting uploading, 312 uploading, 309 photos. See images photos table designing, 345 displaying resized, 392-396 inserting data in, 368, 369-371 structuring, 346, 347-348 updating, 409-413 PHP (PHP: Hypertext Preprocessor) about, 5-6, 8 applying/removing comment tags, 13-14 arithmetic operators for, 91-96 arrays, 70-73 autocompletion of defined variables, 20-21 automating repetitive tasks, 96-100 balancing braces, 12 benefits/disadvantages of, 7-8 code blocks above DOCTYPE declarations, 108, 181

code collapse, 12–13 comments in scripts, 62 conditional statements, 74-79 directives configuring PHP Core, 33 embedding in page, 61-62 enabling Live Code, 14-15 ending statements with semicolons, 63 error messages in, 108–109 features in previous Dreamweaver versions, 9–15 finding arguments for functions, 82-84 functions, 79-88 improved features for, 4, 15-25 including external files with, 101–108 line numbering, 10 mail() function in, 259-260 objects and resources in, 88-91 opening and closing tags, 61 programming techniques in, 61 real-time syntax checking, 21 refreshing page elements with jQuery and, 450-457 regular expressions in, 233 removing error messages from deployed scripts, 465-467 scripts and case sensitivity, 106 security measures in, 34 sending email in, 258, 259-263 sessions in, 188 Split Code view for, 13 syntax coloring, 10–12 timestamps in, 155, 335 uploading files with, 305 using databases with, 8–9 values considered false in, 79 variables, 63–69 weakly typed language, 64–65 when browsers request web page using, 6 PHP testing environments about, 27 activating configuration changes, 39 adding subfolder to server root for, 48 changing php.ini file, 36–39 checking configuration of, 36 creating site definition, 48-52 defining multiple servers, 52–53 enabling other PHP extensions, 38-39 installing software for, 28 level of error reporting, 38 locating web server's document root, 30 managing site definitions, 54–56 requirements for, 28 sending feedback by email, 273–276 setting temporary upload directory, 35 setting up, 28–29 setting up PHP site in Dreamweaver, 47-54 testing, 31–32, 53–54 troubleshooting servers, 54 understanding configuration page, 32–33

using magic_quotes-gpc, 35 virtual hosts, 40-47 WordPress installation in, 115-118 Zend Mail connections tested in, 266 php.ini file activating configuration changes in, 39 backing up, 37 editing, 37 loading old versions of, 32 making changes to, 36-39 recommended settings for, 36 phpMyAdmin. See also MySQL; PHP choosing as MySQL UI, 141, 142 creating user account from, 162 editing database-specific privileges, 163-164 saving users table in, 221 setting up MySQL database from, 113-114 transferring data to remote server using, 462-464 places table adding details in, 358-363 creating management page for, 385–390 deleting records from, 405-408 inserting data in, 368-369, 370-371 structuring, 346-347 updating records in, 396-405 using state_id as foreign key, 354-358 places2photos table designing, 345-346 inserting data in, 368-371 plain authentication for Zend_Mail, 261 planning CMSs, 345-346 assessing task for, 345-346 developing database structure for, 346-348 POST method, 73 \$_POST superglobal array, 71–73, 274 effect on form elements, 277-280 handling checkboxes, 280-282 multiple-choice <select> lists, 285-286 radio button group handling, 282-284 single-choice <select> menus, 284-285 Preferences panel changing default options in, 10 configuring dynamically related files, 22-23 primary keys defined, 145 inserting as foreign key, 354-358 linking tables using, 345 organizing table columns for, 147 planning, 347, 348 removing deleted record references to, 408 retrieving record details for updating, 396-397 privileges editing database-specific, 163-164 MySQL read/write, 161–164 security hazards of root user accounts, 161 properties, 88 public files, 475

Q

queries. See also SELECT queries displaying results from server behavior's, 194–196 looping through multiple results of, 196–197 paging through results with Zend_Paginator, 383–385 rewriting URLs without query string, 457–458 syntax of SELECT query, 379 using variable in SELECT, 244–245 quotation marks array variables with, 68–69 escape characters used in double-quoted strings, 69 magic quotes, 35, 237–238 replacing single with double, 356 variables with, 67–68 quoteInto() method, 245

R

radio buttons creating, 326-329 using in forms, 277, 278, 280, 282-284 Radio Group dialog box, 326-327 read/write privileges for MySQL, 161-164 reCAPTCHA about, 264, 265 incorporating widget in validation script, 270-272 Record Insertion Form Wizard, 177 records confirming deletion of, 204-206 deleting, 385, 405-408 displaying results of multiple loops through database, 196-197 displaying selected, 194–196 inserting into table, 177-182 links to specific, 198-200 notifying if none in table, 206-207 organization of, 144 retrieving, 192-194 selecting with SQL, 379-383 updating, 200-204, 385, 396-405 Recordset dialog box, 193, 194, 200-202 Recordset Navigation Bar, 207 Recordset server behavior displaying selected records, 194–196 loading record into update form, 200-202 selecting records, 192–194 Redisplay Checkbox server behavior, 282 Redisplay on Error dialog box, 242, 267 Redisplay Text Area dialog box, 269 refreshing page content onchange event handler for images, 450-456 script refreshing places, 456–457 using Ajax, 418, 419 without reloading using jQuery, 448-450 register_globals directive, 34

registering virtual hosts in IIS, 44-45 Windows virtual hosts in Apache, 42-45 registration forms building validation scripts for, 230-236 Check New Username server behavior for, 185 checking for duplicate usernames, 245-246 controlling user access with Zend_Auth, 248-249 creating custom behaviors for, 238-241 dealing with unwanted backslashes, 237-238 handling errors with finished, 242-244 inserting records in, 177-182 preserving input when validation fails, 236-237 resetting forgotten passwords, 286-299 testing, 182-183 validation and logic for, 218-219 Zend Framework improvements to, 218–223 regular expressions in PHP, 233 reiectina invalid file types, 314 oversize files, 314-318 relational databases about, 144 cross-referencing tables in, 345-346 relative links, 251 relative path, 101 remote servers accessing control panel on, 465 creating folder for file uploads, 469-470 file transfers outside site root, 475 moving files to and from, 473-474 setting permissions for file uploads, 469-470 setting up FTP access in Dreamweaver for, 470-475 transferring data to, 462-465 using magic_quotes-gpc on, 35 removing comment tags, 13-14 images from CMSs, 414 server behaviors, 182 renaming duplicate files, 321–326 Repeat Region server behavior, 196–197 Repository view, 474 resetting forgotten passwords, 286-299 building reset request script, 288–293 building script for, 294-299 updating records with Zend_Db, 286-299 resources, 90-91 restoring site definitions, 56 Restrict Access To Page server behavior, 187-188 restyling WordPress site, 111-139 activating child theme, 122-127 adding media files, 126-127 developing themes and child themes, 119–120 editing page template, 136-139 enabling site-specific code hints, 135

file structure for, 112 installing WordPress in testing environment, 115–118 MySQL database and user account setup, 113–114 preparing file for, 120–122 styling and redesigning CMSs, 111 rewriting URLs without query string, 457–458 root user account deleting in MAMP, 162 privileges and security hazards of, 161 rules inspecting in Code Navigator, 134 revising, 132–133 Rules pane (CSS Styles panel), 129

S

saving multiple columns, 157-159 table definition, 159 uploaded images, 368 users table in phpMyAdmin, 221 scripts. See also Zend Framework adapting mail processing, 336–338 adjusting file paths, user accounts, and passwords, 467-468 arguments in, 80 building ZF validation, 230–236 comments in, 62 creating Zend_Auth login, 249–251 developing uploading, 309-311 editing PHP configuration to repair, 32–33 eliminating errors in, 34 handling ZF exceptions with try and catch, 226-227 including external scripts in PHP, 105–108 incorporating reCAPTCHA widget in validation, 270-272 inserting image details in table with, 358–363 modifying uploading to sending attached files, 334-340 overwriting files during uploading, 305, 326-329 preventing renaming of files by, 328 providing error messages for login, 249-251 removing error messages from, 465-467 renaming files with duplicate names, 321–326, 366-367 reset request, 288-293 resetting passwords with, 294-299 validating file uploading, 312–318 security hosting company limits on MySQL user accounts, 165 importance of, 34 replacing WordPress values and files for, 116 using view-only user accounts for web connections, 161, 165

SELECT queries creating with Recordset server behavior, 192 displaying photos on page with, 398 executing with fetchAll() or fetchRow(), 290 INNER JOIN selections of images and places, 394-395 selecting all records with, 379 sorting returns in, 380-381 specifying search criteria with WHERE clause, 379-380 syntax of, 379 using variable in, 244-245 <select> menus inserting photos separately via, 372-373 populating dynamically with server behavior, 354-358 single-choice, 225, 284-285 Server Behavior Builder about, 212 creating server behaviors, 238-241 illustrated, 240 server behaviors, 170-209 about, 170-171 checkboxes redisplayed after form errors, 281-282 connecting to database before using, 171 converting Zend_Paginator into, 390-392 creating custom, 238-241 creating links to specific records, 198-200 defined, 4 Delete Record, 204-206 encrypting passwords with, 190-192 failing to load next page, 184 finding and selecting records, 192–194 inserting records into table, 177–182 inserting state_id as foreign key, 354–358 Log In User, 185-187, 190-192, 249 Log Out User, 188-190 modifying error displays for email, 267-269 pros and cons of, 207-209 radio buttons redisplayed after errors, 282-284 removing, 182 Repeat Region, 196–197 replacing single quotation marks with double, 356 Restrict Access To Page, 187–188 Show Region, 206-207 testing registration form, 182-183 types of User Authentication, 183, 185 updating database records, 200-204 server root adding subfolder to, 48 indicating URL port location in MAMP, 30 server-side technology. See also Apache web servers; PHP testing environments; web servers function of, 5-6 PHP's advantage/disadvantage as, 7-8 validating input on servers, 212, 227-229

sessions starting in upload script, 334-335 tracking variables in scripting, 295 using, 188 SET data type, 152–154 Show Region server behavior, 206-207 sidebar placement, 131-132 single-choice <select> menus, 225, 284-285 singleton objects, 248 site definitions backing up and restoring, 56 creating, 48-52 editing, 54–56 site root adding code hints for CMS in, 135 creating internal links to, 40 nonstandard functions for link to, 55 transferring deployed files outside, 475 Site Setup dialog box, 49–52 site-specific code hinting about, 18 Drupal, Joomla, and WordPress, 135 Zend Framework, 215–218 Site-Specific Code Hints dialog box, 135 software for test environments, 27, 28 sorting collation and, 156-157 SELECT query returns, 380-381 spam parameters preventing, 259, 260 stopping with CAPTCHA, 263-265 Split Code view checking code insertion, 285 using, 13, 270 spreadsheets, relational databases vs., 143-144 Spry framework, 421 SQL, 379–383. See also MySQL; queries; SELECT queries joining tables, 381-382 naming practices for, 149–150 pronunciation of, 150 selecting all records in table, 379 using functions with Zend_Db, 288 .sql files importing data from, 165-167 populating exercise tables with, 419 transferring data to remote database, 461, 462-464 SQLyoq choosing as MySQL UI, 142 transferring data to remote database using, 464-465 SSH (Secure Shell) account access, 464 statements defined, 63 ending with semicolon, 63

strings assigning to variable, 66 defined, 66 escape characters in, 69 joining, 94-95 using in calculations, 93-94 style sheets. See also restyling WordPress site; style. css file adjusting theme, 128-134 included in child theme, 120 inspecting rules in Code Navigator, 134 moving sidebar from right to left, 131–132 revising rules, 132–133 style.css file about, 120 adding comments to, 121-122 code introducing, 121 superglobal arrays, 71-73 synchronizing site files, 472 syntax PHP syntax coloring, 10–12 real-time checking of, 21 SELECT query, 379

T

tables adding photos separately to, 371-374 cross-referencing, 345 Crow's Foot notation for, 145–146 data insertion in multiple, 342-375 defining users, 157–159, 161 deleting data with DROP TABLE option, 462 foreign and primary keys for, 145, 147-148 inserting data in, 368-371 inserting records into, 177-182 joining, 381-382 many-to-many relationship for, 147 naming, 149-150 notifying if no records in, 206-207 one-to-many relationship for, 145, 146, 147 organizing primary and foreign key columns on, 147 planning structure for new CMS, 346-348 primary key used as foreign key in, 354-358 query selecting all records in, 379 relationship between, 144 retrieving list of records from, 192–194 viewing definition files for, 175 Tag selector, selecting HTML elements using, 196 tags applying/removing comment, 13-14 omitting closing, 224 opening and closing PHP, 61 templates. See also themes editing WordPress, 136-139 styling WordPress child themes, 128-134 Twenty Ten theme, 120

testing. See also PHP testing environments email attachment script., 339–340 HTML replacement script, 455 multiple criteria with chained validators, 230 removing spaces from filenames, 320-321 submission of forms with \$_POST, 277 Zend_Auth login scripts, 251–252 Zend_File uploads, 308–309 TEXT data type, 151–152 TextWrangler, 40, 46 themes. See also Twenty Ten theme developing child, 119-120 files within, 120 placing images on Dashboard for, 122 styling child, 128-134 throwing exceptions. See errors thumbnails for LightBox Gallery Widget, 437 TIME data type, 154 time zones, 89-90 timestamps TIMESTAMP data type, 154, 155 using PHP, 335 transactions, 160 transferring data to remote database commercial tools for, 464-465 removing error messages from scripts, 465-467 using remote server's control panel, 465 using .sql file for, 461, 462-464 troubleshooting errors during database connection, 226 missing email, 276 MySQL connections, 174–175 server behavior fails to load next page, 184 testing servers, 54 uploading, 311-312 web server installations, 32 TRUE keyword, 65 TRUE value, 75-76, 79 try and catch blocks, 226-227 turning on/off code hinting, 19–20 display_errors directive, 34, 77 highlighting in Design view, 181 magic_quotes-gpc directive controls, 35 syntax coloring, 11 Twenty Ten theme Header link in, 124 negative right margin on <div>, 132 overriding functions.php in, 124-126 style.css code for, 121 templates in, 120

U

unique index, 219–220 unsubscribing registered users, 299–300 update() method, 286–287 Update Photo Details page, 409-413 Update Place page, 396-405 Update Record server behavior, 202-204 updating database records, 192, 200-204 images on CMS page, 395 overview of record, 385 photos table, 409-413 places table, 396-405 records using associative arrays, 287 retrieving record details for primary key, 396–397 using Zend_Db, 286-299 upgrades with HTML5 Pack, 118 uploading files, 302-341 adding Zend Framework to remote server, 468–469 configuring destination for, 310 creating basic script for, 309-311 customizing error messages for, 332-333 designing form for, 306-308 displaying validation messages while, 314-318 files with PHP, 305 handling multiple files, 329-332 maximum file size when, 306, 312 overwriting when, 305, 326-329 password-protection and, 320 photos for CMSs, 363-368 removing filename spaces during, 318-321 renaming duplicates while, 321-326, 366-367 sending attached files, 334-340 to temporary PHP directory, 35 testing scripts locally for, 308-309 troubleshooting, 311-312 using PHP, 305 validating while, 312-314, 363-368 Zend_File for, 308-333 URLs removing spaces from, 318–321 rewriting without query string, 457-458 setting remote server, 471 user accounts adjusting in deployed site scripts, 467-468 defining users table for user registration, 157-159, 161 read/write privileges for, 161-164 WordPress, 113-114 User Authentication server behaviors Check New Username, 185 Log In User, 185-187, 190-192, 249 Log Out User, 188-190 Restrict Access To Page, 187-188 types of, 183, 185 ZF authentication vs., 247 user registration. See registration forms users. See also passwords; user accounts authenticating with Zend_Auth component, 247-255

checking usernames, 185–187, 245–246 displaying details of logged-in, 254–255 logging in, 185-187 logging out, 188–190 notifying if no records in tables, 206-207 resetting forgotten passwords, 286-299 storing username in variable, 188 tracking with cookies, 188 unsubscribing registered, 299–300 validating input on server, 212, 227-229 users table adding extra columns to, 220-221 adding unique index to, 219–220 columns needed in, 157 defining, 157–159, 161 defining for user registration, 157–159, 161 UTF-8 encoding, 262

V

validating input, 212–255. See also Zend_Validate component building validation scripts, 230-236 checkboxes, 280 email addresses, 273-274 file uploading, 312–314 file validation classes, 313 messages confirming upload file, 314-318 photos uploaded to CMSs, 363-368 preserving input when validation fails, 236-237 testing multiple criteria, 230, 234–235 using strong passwords, 235-236 validation error codes, 333 Zend_Validate for, 227-229 values assigning to variables, 64–65 considered false, 79 selecting records marked with lighting bolt, 198 VARCHAR data type, 151–152, 154 variable scope autocompletion of, 20 jQuery, 449 PHP. 87 variables, 63-69 accented characters in names of, 64 array, 68-69, 70-73 assigning values to, 64-65 defined, 61, 63 escape characters in strings, 69 naming, 63-64 objects as, 88 placement of operators with, 92 reassigning results to same, 95–96 text assigned to, 66 tracking session, 295 using in SELECT query, 244-245

using with quotation marks, 67-68

Vertical Split view, 232 viewing table definition files, 175 virtual hosts, 40–47 configuring on Mac OS X, 45–47 creating, 40, 41 getimagesize() difficulties with site-rootrelative path, 446 manually adding on Mac systems, 46–47 multiple websites vs., 40–41 setting up on Windows, 42–45 unsupported on IIS, 41 using, 47

W

WampServer activating configuration changes for, 39 integrating PHP into IIS with, 29 locating document root for web server, 30 using subfolder of server root with, 48 warnings, 108 web pages adding log out link to, 188-190 creating master/detail set, 437-441 effect of PHP code for browsers requesting, 6 ensuring function if JavaScript disabled, 418, 419 failure to load next, 184 hiding part of page when recordset empty, 206-207 inserting widget in, 427-430 password-protecting, 187-188, 253 refreshing without reloading, 448-450 using PHP with, 5-6 web servers. See also Apache web servers; PHP testing environments; validating input activating PHP configuration changes for, 39 creating multiple, 52-53 defined, 28 handling asynchronous requests and responses, 7 locating document root, 30 removing spaces from Linux filenames, 318-321 storing uploaded files on, 334 support for URL rewriting, 457-458 testing and remote, 52-53 tracking logged in users on, 188 understanding PHP configuration for test, 32-33 uploading files to, 304 websites. See also creating PHP sites; deploying sites online; PHP testing environments creating internal links in, 40 location of Connections folder for live, 173 site definitions for, 48-52, 54-56 site root, 40, 55, 135, 475 using virtual hosts vs. multiple, 40-41 WHERE clause, 379-380, 381-382 while loops, 97

Widget dialog box, 428 widgets. See also Adobe Widget Browser; LightBox Gallery widget choosing from Adobe Exchange, 422–424 creating preset for, 425-427 dependent files for, 428, 429 finding, 420-421 inserting in page, 427-430 populating dynamically, 430–437 previewing, 423-424 revising in child themes, 123 Windows OS systems. See also IIS avoiding formatting recipient's address for, 259 Bcc recipient's address visible in, 259 drop shadow rendering in Live View, 134 editing php.ini in, 37 enabling other PHP extensions, 38–39 integrating PHP into IIS, 29 permissions uploading files in IIS, 309 setting remote server permissions for file uploads, 469-470 using Apache for virtual hosts, 41 virtual host setup in, 42-45 WordPress activating child theme, 122-127 adding media files to, 126–127 configuring for use with MySQL database, 113–114 Dashboard for, 117 developing themes and child themes, 119–120 editing page template for, 136–139 enabling site-specific code hints for, 135 installing in PHP testing environment, 115–118 preparing to style child themes, 120-122 replacing values and files when security breached, 116 structure of, 112 styling child themes, 128-134 WPI (Web Platform Installer), 29

Х

 XAMPP

 activating configuration changes for, 39
 integrating PHP into IIS with, 29
 locating document root for web server, 30
 registering Windows-based virtual hosts in Apache, 42, 43
 using subfolder of server root with, 48

 XHTML (Extensible Hypertext Markup Language), 4
 XML (Extensible Markup Language), 7

Υ

YEAR data type, 154, 155

Ζ

Zend, 8 Zend Framework (ZF). See also specific ZF components about, 4, 212, 213-214 adding extra columns to users table, 220-221 adding unique index to users table, 219-220 authenticating user credentials, 247-255 building validation scripts, 230–236 chaining validators to test multiple criteria, 230, 234-235 code hinting for Zend_Auth, 250 components of, 213 controlling user access with, 248-249 database connections using Zend_Db, 223-226 documentation for, 215 handling exceptions with try and catch blocks, 226-227 improving MySQL registration form in, 218–223 input validation in, 227-229 installing, 214-215 loading class files for, 221-223 logging out with Zend_Auth, 253-254 password-protecting pages, 253 preserving input when validation fails, 236–237 querying database in, 244-245 scripting error messages for login, 249-251 site-specific code hints for, 215-218 testing login scripts, 251-252 uploading to remote server, 468-469 User Authentication server behaviors vs. authentication with, 247 using Zend_Mail, 261-263 validating strong passwords, 235-236 Zend Select Menu dialog box, 372 Zend_Auth component, 247–255 controlling user access with, 248-249 logging out with, 253-254 password-protecting pages with, 253 scripting for login error messages, 249-251 selecting code hinting for, 250 testing login scripts, 251-252 using User Authentication server behaviors vs., 247 Zend_Captcha_Figlet class, 263, 264 Zend_Captcha_Image class, 264 Zend_Captcha_ReCaptcha class, 264, 265

Zend_Db component checking for duplicate usernames, 245–246 inserting user details in database, 246-247 updating records with, 286-299 using SQL functions with, 288 Zend_Db object connecting to database with, 223-226 using select() method for, 383 Zend_File component about, 308 creating local folder for uploading test scripts, 308-309 handling multiple file uploads, 329–332 overwriting files during uploading, 305, 326–329 renaming duplicate files, 321–326 uploading files with, 305 validating uploaded files, 312-318 Zend_Mail component creating attachments, 334–340 creating mail connector script, 266 encoding specification for, 262 obtaining public and private reCAPTCHA keys, 265 processing user feedback with, 266-276 sending feedback by email, 273–276 stopping spam with CAPTCHA, 263–265 troubleshooting missing email, 276 using reCAPTCHA in validation script, 270–272 Zend_Mail_Transport_Smtp object, 266 Zend_Paginator component converting to server behavior, 390-392 creating management page for places table, 385-390 dialog box for, 392 navigating with, 383-385 page for managing photos table, 392-396 properties of, 385 Zend_Service_ReCaptcha class, 265, 270, 271 Zend_Validate component building validation scripts, 230-236 chaining validators to test multiple criteria, 230, 234-235 input validation with, 227-229 preserving input when validation fails, 236–237 using variable in SELECT query, 244-245 validating strong passwords, 235-236



Meet Creative Edge.

A new resource of unlimited books, videos and tutorials for creatives from the world's leading experts.

Creative Edge is your one stop for inspiration, answers to technical questions and ways to stay at the top of your game so you can focus on what you do best—being creative.

All for only \$24.99 per month for access—any day any time you need it.



peachpit.com/creativeedge