

Julie C. Meloni

**FIFTH EDITION**

**STARTER KIT**

CD includes a complete  
starter kit for Windows\*,  
Linux\* and Mac\* OS X

Sams **Teach Yourself**

# PHP, MySQL® and Apache

**All**  
in **One**

**SAMS**

Julie C. Meloni

Sams **Teach Yourself**

**PHP,  
MySQL<sup>®</sup>  
Apache** and  
in **All  
One**

**SAMS**

800 East 96th Street, Indianapolis, Indiana, 46240 USA

# **Sams Teach Yourself PHP, MySQL® and Apache All in One**

Copyright © 2012 by Pearson Education, Inc.

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

ISBN-13: 978-0-672-33543-3

ISBN-10: 0-672-33543-3

Library of Congress Cataloging-in-Publication Data

Meloni, Julie C.

Sams teach yourself PHP, MySQL and Apache : all in one / Julie C.

Meloni.

p. cm.

Includes index.

ISBN-13: 978-0-672-33543-3 (pbk. w/cd)

ISBN-10: 0-672-33543-3

1. Web site development. 2. PHP (Computer program language) 3. Apache (Computer file : Apache Group) 4. MySQL (Electronic resource)  
I. Title.

TK5105.888.M45 2012

005.13-dc23

2012016353

Printed in the United States of America

First Printing May 2012

## **Trademarks**

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Sams Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

## **Warning and Disclaimer**

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an “as is” basis. The author and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the CD or programs accompanying it.

## **Bulk Sales**

Sams Publishing offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact

**U.S. Corporate and Government Sales**

**1-800-382-3419**

**corpsales@pearsontechgroup.com**

For sales outside of the U.S., please contact

**International Sales**

**international@pearsoned.com**

## **Acquisitions Editor**

*Mark Taber*

## **Development Editor**

*Songlin Qiu*

## **Managing Editor**

*Sandra Schroeder*

## **Project Editor**

*Mandie Frank*

## **Copy Editor**

*Keith Cline*

## **Indexer**

*Heather McNeill*

## **Proofreader**

*Paula Lowell*

## **Technical Editor**

*Timothy Boronczyk*

## **Publishing Coordinator**

*Vanessa Evans*

## **Media Producer**

*Dan Scherf*

## **Designer**

*Gary Adair*

## **Compositor**

*Studio Galou, LLC*

# Contents at a Glance

Introduction .....	1
<b>PART I: Getting Up and Running</b>	
1 Installation QuickStart Guide with XAMPP .....	5
2 Installing and Configuring MySQL .....	15
3 Installing and Configuring Apache .....	37
4 Installing and Configuring PHP .....	59
<b>PART II: PHP Language Structure</b>	
5 The Building Blocks of PHP .....	75
6 Flow Control Functions in PHP .....	99
7 Working with Functions .....	119
8 Working with Arrays .....	139
9 Working with Objects .....	149
<b>PART III: Getting Involved with the Code</b>	
10 Working with Strings, Dates, and Time .....	159
11 Working with Forms .....	189
12 Working with Cookies and User Sessions .....	213
13 Working with Files and Directories .....	229
14 Working with Images .....	261
<b>PART IV: PHP and MySQL Integration</b>	
15 Understanding the Database Design Process .....	283
16 Learning Basic SQL Commands .....	297
17 Using Transactions and Stored Procedures in MySQL .....	349
18 Interacting with MySQL Using PHP .....	357

## **PART V: Basic Projects**

<b>19</b>	Managing a Simple Mailing List .....	373
<b>20</b>	Creating an Online Address Book .....	387
<b>21</b>	Creating a Simple Discussion Forum .....	417
<b>22</b>	Creating an Online Storefront .....	437
<b>23</b>	Creating a Shopping Cart Mechanism .....	451
<b>24</b>	Creating a Simple Calendar .....	467
<b>25</b>	Restricting Access to Your Applications .....	491
<b>26</b>	Logging and Monitoring Web Server Activity .....	509
<b>27</b>	Application Localization .....	527
<b>28</b>	Working with XML and JSON .....	541

## **PART VI: Administration and Fine-Tuning**

<b>29</b>	Apache Performance Tuning and Virtual Hosting .....	555
<b>30</b>	Setting Up a Secure Web Server .....	573
<b>31</b>	Optimizing and Tuning MySQL .....	589
<b>32</b>	Performing Software Upgrades .....	605
<b>33</b>	Using Application Frameworks .....	611
	Index .....	619

# Table of Contents

Introduction	1
<b>PART I: Getting Up and Running</b>	
<b>CHAPTER 1: Installation QuickStart Guide with XAMPP</b>	<b>5</b>
Using Third-Party Installation Packages	5
Installing XAMPP on Linux/UNIX	6
Installing XAMPP on Windows	8
Installing XAMPP on Mac OS X	11
Securing XAMPP	13
Troubleshooting	14
<b>CHAPTER 2: Installing and Configuring MySQL</b>	<b>15</b>
Current and Future Versions of MySQL	15
How to Get MySQL	16
Installing MySQL on Linux/UNIX	16
Installing MySQL on Mac OS X	18
Installing MySQL on Windows	20
Troubleshooting Your Installation	26
Basic Security Guidelines	27
Introducing the MySQL Privilege System	28
Summary	33
Q&A	34
Workshop	34
<b>CHAPTER 3: Installing and Configuring Apache</b>	<b>37</b>
Current and Future Versions of Apache	37
Choosing the Appropriate Installation Method	38
Installing Apache on Linux/UNIX	39
Installing Apache on Mac OS X	42
Installing Apache on Windows	42
Apache Configuration File Structure	45
Apache Log Files	50
Apache-Related Commands	51
Starting Apache for the First Time	53

Troubleshooting .....	55
Summary .....	56
Q&A .....	56
Workshop .....	57
<b>CHAPTER 4: Installing and Configuring PHP</b> .....	<b>59</b>
Current and Future Versions of PHP .....	59
Building PHP on Linux/UNIX with Apache .....	60
Installing PHP on Mac OS X .....	63
Installing PHP on Windows .....	63
php.ini Basics .....	65
Testing Your Installation .....	65
Getting Installation Help .....	66
The Basics of PHP Scripts .....	67
Summary .....	73
Q&A .....	73
Workshop .....	74
<b>PART II: PHP Language Structure</b>	
<b>CHAPTER 5: The Building Blocks of PHP</b> .....	<b>75</b>
Variables .....	75
Data Types .....	78
Operators and Expressions .....	85
Constants .....	94
Summary .....	96
Q&A .....	96
Workshop .....	96
<b>CHAPTER 6: Flow Control Functions in PHP</b> .....	<b>99</b>
Switching Flow .....	99
Loops .....	105
Code Blocks and Browser Output .....	114
Summary .....	116
Q&A .....	116
Workshop .....	116

<b>CHAPTER 7: Working with Functions</b>	<b>119</b>
What Is a Function? .....	119
Calling Functions .....	120
Defining a Function .....	121
Returning Values from User-Defined Functions .....	124
Variable Scope .....	125
Saving State Between Function Calls with the static Statement .....	128
More About Arguments .....	130
Testing for the Existence of a Function .....	133
Summary .....	135
Q&A .....	135
Workshop .....	136
<b>CHAPTER 8: Working with Arrays</b>	<b>139</b>
What Are Arrays? .....	139
Creating Arrays .....	140
Some Array-Related Constructs and Functions .....	144
Summary .....	146
Q&A .....	146
Workshop .....	147
<b>CHAPTER 9: Working with Objects</b>	<b>149</b>
Creating an Object .....	150
Object Inheritance .....	155
Summary .....	157
Q&A .....	157
Workshop .....	157
<b>PART III: Getting Involved with the Code</b>	
<b>CHAPTER 10: Working with Strings, Dates, and Time</b>	<b>159</b>
Formatting Strings with PHP .....	160
Investigating Strings in PHP .....	169
Manipulating Strings with PHP .....	173
Using Date and Time Functions in PHP .....	179
Other String, Date, and Time Functions .....	186



Summary .....	186
Workshop .....	186
<b>CHAPTER 11: Working with Forms</b>	<b>189</b>
Creating a Simple Input Form .....	189
Accessing Form Input with User-Defined Arrays .....	191
Combining HTML and PHP Code on a Single Page .....	194
Using Hidden Fields to Save State .....	197
Redirecting the User .....	198
Sending Mail on Form Submission .....	200
Creating the Form .....	201
Creating the Script to Send the Mail .....	202
Working with File Uploads .....	206
Summary .....	210
Q&A .....	210
Workshop .....	211
<b>CHAPTER 12: Working with Cookies and User Sessions</b>	<b>213</b>
Introducing Cookies .....	213
Setting a Cookie with PHP .....	215
Deleting a Cookie with PHP .....	217
Session Function Overview .....	217
Starting a Session .....	218
Working with Session Variables .....	219
Destroying Sessions and Unsetting Variables .....	223
Using Sessions in an Environment with Registered Users .....	224
Summary .....	225
Q&A .....	226
Workshop .....	226
<b>CHAPTER 13: Working with Files and Directories</b>	<b>229</b>
Including Files .....	229
Using include_once .....	233
Validating Files .....	234
Creating and Deleting Files .....	238
Opening a File for Writing, Reading, or Appending .....	238

Reading from Files .....	239
Writing or Appending to a File .....	245
Working with Directories .....	248
Opening Pipes to and from Processes Using popen() .....	251
Running Commands with exec() .....	254
Running Commands with system() or passthru() .....	255
Summary .....	257
Q&A .....	257
Workshop .....	258
<b>CHAPTER 14: Working with Images</b> .....	<b>261</b>
Understanding the Image-Creation Process .....	261
Necessary Modifications to PHP .....	262
Drawing a New Image .....	263
Modifying Existing Images .....	271
Image Creation from User Input .....	273
Using Images Created by Scripts .....	278
Summary .....	280
Q&A .....	281
Workshop .....	281
<b>PART IV: PHP and MySQL Integration</b>	
<b>CHAPTER 15: Understanding the Database Design Process</b> .....	<b>283</b>
The Importance of Good Database Design .....	283
Types of Table Relationships .....	284
Understanding Normalization .....	289
Following the Design Process .....	292
Summary .....	293
Q&A .....	294
Workshop .....	294
<b>CHAPTER 16: Learning Basic SQL Commands</b> .....	<b>297</b>
Learning the MySQL Data Types .....	298
Learning the Table-Creation Syntax .....	301
Using the INSERT Command .....	302

Using the SELECT Command .....	304
Using WHERE in Your Queries .....	308
Selecting from Multiple Tables .....	310
Using the UPDATE Command to Modify Records .....	316
Using the REPLACE Command .....	319
Using the DELETE Command .....	320
Frequently Used String Functions in MySQL .....	322
Using Date and Time Functions in MySQL .....	331
Summary .....	343
Q&A .....	345
Workshop .....	346
<b>CHAPTER 17: Using Transactions and Stored Procedures in MySQL</b> .....	<b>349</b>
What Are Transactions? .....	349
What Are Stored Procedures? .....	353
Summary .....	355
Q&A .....	355
Workshop .....	356
<b>CHAPTER 18: Interacting with MySQL Using PHP</b> .....	<b>357</b>
MySQL or MySQLi Functions? .....	357
Connecting to MySQL with PHP .....	358
Working with MySQL Data .....	361
Summary .....	369
Q&A .....	370
Workshop .....	370
<b>PART V: Basic Projects</b>	
<b>CHAPTER 19: Managing a Simple Mailing List</b> .....	<b>373</b>
Developing the Subscription Mechanism .....	374
Developing the Mailing Mechanism .....	381
Summary .....	384
Q&A .....	385
Workshop .....	385

<b>CHAPTER 20: Creating an Online Address Book</b>	<b>387</b>
Planning and Creating the Database Tables .....	387
Creating an Include File for Common Functions .....	390
Creating a Menu .....	391
Creating the Record-Addition Mechanism .....	392
Viewing Records .....	398
Creating the Record-Deletion Mechanism .....	404
Adding Subentries to a Record .....	406
Summary .....	414
Q&A .....	414
Workshop .....	414
<b>CHAPTER 21: Creating a Simple Discussion Forum</b>	<b>417</b>
Designing the Database Tables .....	417
Creating an Include File for Common Functions .....	418
Creating the Input Forms and Scripts .....	419
Displaying the Topic List .....	423
Displaying the Posts in a Topic .....	426
Adding Posts to a Topic .....	430
Summary .....	433
Q&A .....	434
Workshop .....	434
<b>CHAPTER 22: Creating an Online Storefront</b>	<b>437</b>
Planning and Creating the Database Tables .....	437
Displaying Categories of Items .....	441
Displaying Items .....	445
Summary .....	448
Q&A .....	448
Workshop .....	448
<b>CHAPTER 23: Creating a Shopping Cart Mechanism</b>	<b>451</b>
Planning and Creating the Database Tables .....	451
Integrating the Cart with Your Storefront .....	453
Payment Methods and the Checkout Sequence .....	462

Summary .....	465
Q&A .....	465
Workshop .....	465
<b>CHAPTER 24: Creating a Simple Calendar</b> .....	<b>467</b>
Building a Simple Display Calendar .....	467
Creating a Calendar Library .....	483
Summary .....	489
Q&A .....	489
Workshop .....	489
<b>CHAPTER 25: Restricting Access to Your Applications</b> .....	<b>491</b>
Authentication Overview .....	491
Apache Authentication Module Functionality .....	493
Using Apache for Access Control .....	497
Combining Apache Access Methods .....	500
Limiting Access Based on HTTP Methods .....	501
Restricting Access Based on Cookie Values .....	501
Summary .....	507
Q&A .....	507
Workshop .....	508
<b>CHAPTER 26: Logging and Monitoring Web Server Activity</b> .....	<b>509</b>
Standard Apache Access Logging .....	509
Standard Apache Error Logging .....	515
Managing Apache Logs .....	517
Logging Custom Information to a Database .....	519
Summary .....	523
Q&A .....	524
Workshop .....	524
<b>CHAPTER 27: Application Localization</b> .....	<b>527</b>
About Internationalization and Localization .....	527
About Character Sets .....	528
Environment Modifications .....	529
Creating a Localized Page Structure .....	531

Localizing Your Application with gettext() .....	536
Summary .....	537
Q&A .....	538
Workshop .....	538
<b>CHAPTER 28: Working with XML and JSON</b> .....	<b>541</b>
What Is XML? .....	541
Accessing XML in PHP Using DOM Functions .....	544
Accessing XML in PHP Using SimpleXML Functions .....	546
Working with JSON .....	549
Summary .....	553
Q&A .....	553
Workshop .....	554
<b>PART VI: Administration and Fine-Tuning</b>	
<b>CHAPTER 29: Apache Performance Tuning and Virtual Hosting</b> .....	<b>555</b>
Performance and Scalability Issues .....	555
Load Testing with ApacheBench .....	559
Proactive Performance Tuning .....	561
Preventing Abuse .....	563
Implementing Virtual Hosting .....	564
Summary .....	569
Q&A .....	570
Workshop .....	571
<b>CHAPTER 30: Setting Up a Secure Web Server</b> .....	<b>573</b>
The Need for Security .....	573
The SSL Protocol .....	574
Obtaining and Installing SSL Tools .....	579
Managing Certificates .....	582
SSL Configuration .....	585
Summary .....	586
Q&A .....	586
Workshop .....	586

<b>CHAPTER 31: Optimizing and Tuning MySQL</b>	<b>589</b>
Building an Optimized Platform .....	589
Benchmarking Your Database Server .....	590
MySQL Startup Options .....	591
Optimizing Your Table Structure .....	593
Optimizing Your Queries .....	594
Using the FLUSH Command .....	595
Using the SHOW Command .....	596
Summary .....	603
Q&A .....	603
Workshop .....	604
<b>CHAPTER 32: Performing Software Upgrades</b>	<b>605</b>
Staying in the Loop .....	605
Upgrading MySQL .....	607
Upgrading Apache .....	608
Upgrading PHP .....	609
Summary .....	610
Workshop .....	610
<b>CHAPTER 33: Using Application Frameworks</b>	<b>611</b>
Understanding Application Frameworks .....	611
Using the MVC Pattern .....	612
Installing and Using PHP Application Frameworks .....	614
Summary .....	617
Workshop .....	617
Activities .....	618
<b>Index</b>	<b>619</b>

## About the Author

**Julie C. Meloni** is a technical consultant who has been developing web-based applications since the Web first saw the light of day. She has authored numerous books and articles on web-based programming and scripting languages and database topics, and you can find translations of her work in 18 different languages. She blogs at [thickbook.com](http://thickbook.com) and [nerdtripping.com](http://nerdtripping.com)—the latter reserved for tips and tricks for traveling while nerdy.

## Acknowledgments

The Apache Software Foundation, the PHP Group, and MySQL AB deserve much more recognition than they ever get for creating these super products that drive the vast majority of the Web.

Although this book is several editions removed from the original text by Daniel Lopez (author of *Sams Teach Yourself Apache 2 in 24 Hours*) and Matt Zandstra (author of *Sams Teach Yourself PHP in 24 Hours*), this book would not exist without their work oh so many years ago.



## **We Want to Hear from You!**

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

You can email or write directly to let us know what you did or didn't like about this book—as well as what we can do to make our books stronger.

*Please note that we cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail we receive, we might not be able to reply to every message.*

When you write, please be sure to include this book's title and author as well as your name and phone number or email address.

E-mail:                    [feedback@sampublishing.com](mailto:feedback@sampublishing.com)

Mail:                      Reader Feedback  
                              Sams Publishing  
                              800 East 96th Street  
                              Indianapolis, IN 46240 USA

## **Reader Services**

Visit our website and register this book at [informit.com/register](http://informit.com/register) for convenient access to any updates, downloads, or errata that might be available for this book.

# Introduction

Welcome to *Sams Teach Yourself PHP, MySQL, and Apache All in One, Fifth Edition*. I'm happy to report that the PHP language and its community of developers and users continues to grow every day—hence the need for a refresh of this book.

Since the previous edition of this book, the “end of life” of PHP 4 finally set in; with the help of a GoPHP5 initiative, web hosting providers and application developers migrated their services and code away from PHP 4—specific features and coding practices and into the world of PHP 5—full of speed and an even greater feature set. As with the previous edition, all the code in this edition is based on the latest version of PHP available at the time of this writing (5.4.0, in this case).

Some of you might have heard of PHP 6 or have seen books touting PHP 6 as the core language used. Well, a version of the language called PHP 6 never materialized—the functionality planned for a version 6 release was added to PHP 5.3 and PHP 5.4. So, have no fear; you're not missing anything if you hear PHP 6 and cannot find anything about it online or at the PHP.net website.

Over the course of this book, you learn the concepts necessary for configuring and managing the Apache web server, the basics of programming in PHP, and the methods for using and administering the MySQL relational database system. The overall goal of the book is to provide you with the foundation you need to understand how seamlessly these technologies integrate with one another and to give you practical knowledge of how to integrate them into functioning websites and web applications. This book should be a first step—not your only step—to more advanced site development.

## Who Should Read This Book?

This book is geared toward individuals who possess a general understanding of the concepts of working in a web-based development environment, be it Linux/UNIX, Windows, or Mac OS X. Installation and configuration instructions assume that you have familiarity with your operating system and the basic methods of building (on Linux/UNIX systems) or installing (on Windows and Mac OS X systems) software.

The lessons that delve into programming with PHP assume no previous knowledge of the language. However, if you have experience with other programming languages, such as ASP (Active Server Pages), JSP (JavaServer Pages), Ruby, or Perl, you will find the going much easier because of your familiarity with such programming elements as variables, control structures, functions, objects, and the like. Similarly, if you have worked with other databases, such as Oracle or Microsoft SQL Server, you already possess a solid foundation for working through the MySQL-related lessons.

The only real requirement is that you already understand static web content creation with HTML. If you are just starting out in the world of web development, you will still be able to use this book, but you should consider working through an HTML tutorial. If you are comfortable creating basic pages, you will be fine.

## How This Book Is Organized

This book is divided into six parts, corresponding to particular topic groups. You should read the chapters within each part one right after another, with each chapter building on the information found in those before it:

- ▶ Part I, “Getting Up and Running,” provides a quick-start guide to installation and walks you through the installation and configuration of MySQL, Apache, and PHP in depth. You need to complete at least one version of these instructions—either the quick-start installation or the longer instructions—before moving on *unless you already have access to a working installation of these technologies through a hosting provider*. Even if you do not need to install and configure MySQL, Apache, and PHP in your development environment, you should still skim these lessons so that you understand the basics of their interaction.
- ▶ Part II, “PHP Language Structure,” is devoted to teaching you the basics of the PHP language, including structural elements such as arrays and objects. The examples will get you in the habit of writing code, uploading it to your server, and testing the results.
- ▶ Part III, “Getting Involved with the Code,” consists of chapters that cover intermediate-level application development topics, including working with forms and files, restricting access, and completing other small projects designed to introduce a specific concept.

- ▶ Part IV, “PHP and MySQL Integration,” contains chapters devoted to working with databases in general, such as database normalization, as well as using PHP to connect to and work with MySQL. Included is a basic SQL primer, which also includes MySQL-specific functions and other information.
- ▶ Part V, “Basic Projects,” consists of chapters devoted to performing a particular task using PHP and MySQL, integrating all the knowledge gained so far. Projects include an address book, a discussion forum, and a basic online storefront, among others. These examples are built in a black-and-white environment, meaning the aesthetic display is minimal. This allows you to focus on the programming and logic involved in building the structures rather than making these items aesthetically pleasing.
- ▶ Part VI, “Administration and Fine-Tuning,” is devoted to administering and tuning Apache and MySQL. It also includes information on virtual hosting and setting up a secure web server.

If you find that you are already familiar with a topic, you can skip ahead to the next chapter. However, in some instances, chapters refer to specific concepts learned in previous chapters, so be aware that you might have to skim a skipped chapter so that your development environment remains consistent with the book.

At the end of many chapters, a few quiz questions test how well you’ve learned the material. Additional activities provide another way to apply the information learned in the chapter and guide you toward using this newfound knowledge in the next chapter.

## About the Book’s Source Code

All the code that appears in listings throughout the chapters is also available on the accompanying CD-ROM. You may also download the source code bundle from the author’s website at <http://www.thickbook.com/>.

Typing the code on your own provides useful experience in making typos, causing errors, and performing the sometimes mind-numbing task of tracking down errant semicolons. However, if you want to skip that lesson and just upload the working code to your website, feel free!

## Conventions Used in This Book

This book uses different typefaces to differentiate between code and plain English and to help you identify important concepts. Throughout the chapters, code, commands, and text you type or see onscreen appear in a `computer` typeface. New terms appear in italics at the point in the text where they are defined. In addition, icons accompany special blocks of information:

### **NOTE**

A Note presents an interesting piece of information related to the current topic.

### **TIP**

A Tip offers advice or teaches an easier method for performing a task.

### **CAUTION**

A Caution warns you about potential pitfalls and explains how to avoid them.

## CHAPTER 19

# Managing a Simple Mailing List

---

### ***In this chapter, you learn the following:***

- ▶ How to create a subscribe/unsubscribe form and script
- ▶ How to create a front end for sending your message
- ▶ How to create the script that sends your message

This chapter provides the first of several hands-on, small projects designed to pull together your PHP and MySQL knowledge. In this chapter, you learn how to create a managed distribution list that you can use to send out newsletters or anything else to a list of email addresses in a database.

As with all the small sample projects in this book, these projects might not be exactly what you plan to build with your new PHP and MySQL knowledge. However, I cannot stress enough that the concepts and examples shown in this and other projects are similar to those you will encounter when developing any application that uses CRUD functionality (create, read, update, delete).

### **NOTE**

---

The mailing mechanism you use in this chapter is not meant to be a replacement for mailing list software, which is specifically designed for bulk messages. You should use the type of system you build in this chapter only for small lists, fewer than a few hundred email addresses.

## Developing the Subscription Mechanism

You learned in earlier chapters that planning is the most important aspect of creating any product. In this case, think of the elements you need for your subscription mechanism:

- ▶ A table to hold email addresses
- ▶ A way for users to add or remove their email addresses
- ▶ A form and script for sending the message

The following sections describe each item individually.

### Creating the subscribers Table

You really need only one field in the subscribers table: to hold the email address of the user. However, you should have an ID field just for consistency among your tables, and because referencing an ID is much simpler than referencing a long email address in where clauses. So, in this case, your MySQL query would look something like this:

```
CREATE TABLE subscribers (
  id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
  email VARCHAR (150) UNIQUE NOT NULL
);
```

Note the use of UNIQUE in the field definition for email. This means that although id is the primary key, duplicates should not be allowed in the email field either. The email field is a unique key, and id is the primary key.

Log in to MySQL via the command line and issue this query. After creating the table, issue a DESC or DESCRIBE query to verify that the table has been created to your specifications, such as the following:

```
mysql> DESC subscribers;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id    | int(11)       | NO   | PRI | NULL    | auto_increment |
| email | varchar(150)  | NO   | UNI | NULL    |                |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Now that you have a table in your database, you can create the form and script that place values in there.

## Creating an Include File for Common Functions

Although there are only two scripts in this process, some common functions exist between them—namely, the database connection information. To make your scripts more concise in situations like this, take the common functions or code snippets and put them in a file to be included in your other scripts via the `include()` function that you learned about in Chapter 13, “Working with Files and Directories.” Listing 19.1 contains the code shared by the scripts in this chapter.

### LISTING 19.1 Common Functions in an Included File

---

```
1: <?php
2: // function to connect to database
3: function doDB() {
4:     global $mysqli;
5:
6:     //connect to server and select database
7:     $mysqli = mysqli_connect("localhost", "joeuser",
8:         "somepass", "testDB");
9:
10:    //if connection fails, stop script execution
11:    if (mysqli_connect_errno()) {
12:        printf("Connect failed: %s\n", mysqli_connect_error());
13:        exit();
14:    }
15: }
16: // function to check email address
17: function emailChecker($email) {
18:     global $mysqli, $safe_email, $check_res;
19:
20:     //check that email is not already in list
21:     $safe_email = mysqli_real_escape_string($mysqli, $email);
22:     $check_sql = "SELECT id FROM SUBSCRIBERS
23:         WHERE email = ".$safe_email."";
24:     $check_res = mysqli_query($mysqli, $check_sql)
25:         or die(mysqli_error($mysqli));
26: }
27: ?>
```

---

Lines 3–15 set up the first function, `doDB()`, which is simply the database connection function. If the connection cannot be made, the script exits when this function is called; otherwise, it makes the value of `$mysqli` available to other parts of your script.

Lines 17–26 define a function called `emailChecker()`, which takes an input and returns an output—like most functions do. We look at this one in the context of the script, as we get to it in Listing 19.2

Save this file as `ch19_include.php` and place it on your web server. In Listing 19.2, you will see how to include this file when necessary in your scripts.



## Creating the Subscription Form

The subscription form is actually an all-in-one form and script called `manage.php`, which handles both subscribe and unsubscribe requests. Listing 19.2 shows the code for `manage.php`, which uses a few user-defined functions to eliminate repetitious code and to start you thinking about creating functions on your own. The code looks long, but a line-by-line description follows (and a lot of the code just displays an HTML form, so no worries).

### LISTING 19.2 Subscribe and Unsubscribe with `manage.php`

```

1: <?php
2: include 'ch19_include.php';
3: //determine if they need to see the form or not
4: if (!$_POST) {
5:     //they need to see the form, so create form block
6:     $display_block = <<<END_OF_BLOCK
7:     <form method="POST" action="$_SERVER[PHP_SELF]">
8:
9:     <p><label for="email">Your E-Mail Address:</label><br/>
10:    <input type="email" id="email" name="email"
11:        size="40" maxlength="150" /></p>
12:
13:    <fieldset>
14:    <legend>Action:</legend><br/>
15:    <input type="radio" id="action_sub" name="action"
16:        value="sub" checked />
17:    <label for="action_sub">subscribe</label><br/>
18:    <input type="radio" id="action_unsub" name="action"
19:        value="unsub" />
20:    <label for="action_unsub">unsubscribe</label>
21:    </fieldset>
22:
23:    <button type="submit" name="submit" value="submit">Submit</button>
24:    </form>
25: END_OF_BLOCK;
26: } else if (($_POST) && ($_POST['action'] == "sub")) {
27:     //trying to subscribe; validate email address
28:     if ($_POST['email'] == "") {
29:         header("Location: manage.php");
30:         exit;
31:     } else {
32:         //connect to database
33:         doDB();
34:
35:         //check that email is in list
36:         emailChecker($_POST['email']);
37:
38:         //get number of results and do action
39:         if (mysqli_num_rows($check_res) < 1) {
40:             //free result
41:             mysqli_free_result($check_res);
42:
43:             //add record
44:             $add_sql = "INSERT INTO subscribers (email)
45:                 VALUES('".$_safe_email."')";

```

```

46:         $add_res = mysqli_query($mysqli, $add_sql)
47:             or die(mysqli_error($mysqli));
48:         $display_block = "<p>Thanks for signing up!</p>";
49:
50:         //close connection to MySQL
51:         mysqli_close($mysqli);
52:     } else {
53:         //print failure message
54:         $display_block = "<p>You're already subscribed!</p>";
55:     }
56: }
57: } else if (($_POST) && ($_POST['action'] == "unsub")) {
58:     //trying to unsubscribe; validate email address
59:     if ($_POST['email'] == "") {
60:         header("Location: manage.php");
61:         exit;
62:     } else {
63:         //connect to database
64:         doDB();
65:
66:         //check that email is in list
67:         emailChecker($_POST['email']);
68:
69:         //get number of results and do action
70:         if (mysqli_num_rows($check_res) < 1) {
71:             //free result
72:             mysqli_free_result($check_res);
73:
74:             //print failure message
75:             $display_block = "<p>Couldn't find your address!</p>";
76:             $display_block = "<p>No action was taken.</p>";
77:         } else {
78:             //get value of ID from result
79:             while ($row = mysqli_fetch_array($check_res)) {
80:                 $id = $row['id'];
81:             }
82:
83:             //unsubscribe the address
84:             $del_sql = "DELETE FROM subscribers
85:                 WHERE id = ".$id;
86:             $del_res = mysqli_query($mysqli, $del_sql)
87:                 or die(mysqli_error($mysqli));
88:             $display_block = "<p>You're unsubscribed!</p>";
89:         }
90:         mysqli_close($mysqli);
91:     }
92: }
93: ?>
94: <!DOCTYPE html>
95: <html>
96: <head>
97: <title>Subscribe/Unsubscribe to a Mailing List</title>
98: </head>
99: <body>
100: <h1>Subscribe/Unsubscribe to a Mailing List</h1>
101: <?php echo "$display_block"; ?>
102: </body>
103: </html>

```

---

Listing 19.2 might be long, but it's not complicated. In fact, it could be longer were it not for the user-defined functions placed in `ch19_include.php` and included on line 2 of this script.

Line 4 starts the main logic of the script. Because this script performs several actions, you need to determine which action it is currently attempting. If the presence of `$_POST` is false, you know that the user has not submitted the form; therefore, you must show the form to the user.

Lines 6–25 create the subscribe/unsubscribe form by storing a string in the `$display_block` variable using the heredoc format. In the heredoc format, the string delimiter can be any string identifier following `<<<`, as long as the ending identifier is on its own line, as you can see in this example on line 25.

## NOTE

You can learn more about the heredoc and other string formats in the PHP Manual at <http://www.php.net/manual/en/language.types.string.php>.

In the form, we use `$_SERVER[PHP_SELF]` as the action (line 7), and then create a text field called `email` for the user's email address (lines 9–11) and set up a set of radio buttons (lines 13–21) to find the desired task. At the end of the string creation, the script breaks out of the `if...else` construct, skips down to line 101, and proceeds to print the HTML stored in the `$display_block` variable. The form displays as shown in Figure 19.1.

**FIGURE 19.1**  
The subscribe/  
unsubscribe  
form.

The screenshot shows a web browser window with the address bar displaying `http://localhost/19/manage.php`. The page title is "Subscribe/Unsubscribe to a Mailing List". The form contains the following elements:

- A text input field labeled "Your Email Address:".
- A section labeled "Action:" containing two radio buttons: "subscribe" (which is selected) and "unsubscribe".
- A "Submit" button at the bottom of the form.

Back inside the `if...else` construct, if the presence of `$_POST` is true, you need to do something. There are two possibilities: the subscribing and unsubscribing actions for the email address provided in the form. You determine which action to take by looking at the value of `$_POST['action']` from the radio button group.

In line 26, if the presence of `$_POST` is true and the value of `$_POST['action']` is "sub", you know the user is trying to subscribe. To subscribe, the user needs an email address, so check for one in lines 28–30. If no address is present, redirect the user back to the form.

However, if an address is present, call the `doDB()` function (stored in `ch19_include.php`) in line 34 to connect to the database so that you can issue queries. In line 36, you call the second of our user-defined functions: `emailChecker()`. This function takes an input (`$_POST['email']`), in this case and processes it. If you look back to lines 21–25 of Listing 19.1, you'll see code within the `emailChecker()` function that issues a query in an attempt to find an `id` value in the `subscribers` table for the record containing the email address passed to the function. The function then returns the resultset, called `$check_res`, for use within the larger script.

Note the definition of global variables at the beginning of both user-defined functions in Listing 19.1. These variables need to be shared with the entire script, and so are declared global.

**NOTE**

Jump down to line 39 of Listing 19.2 to see how the `$check_res` variable is used: The number of records referred to by the `$check_res` variable is counted to determine whether the email address already exists in the table. If the number of rows is less than 1, the address is not in the list, and it can be added. The record is added, the response is stored in lines 44–48, and the failure message (if the address is already in the table) is stored in line 54. At that point, the script breaks out of the `if...else` construct, skips down to line 101, and proceeds to print the HTML currently stored in `$display_block`. You'll test this functionality later.

The last combination of inputs occurs if the presence of `$_POST` is true and the value of the `$_POST['action']` variable is "unsub". In this case, the user is trying to unsubscribe. To unsubscribe, an existing email address is required, so check for one in lines 59–61. If no address is present, send the user back to the form.

If an address is present, call the `doDB()` function in line 64 to connect to the database. Then, in line 67, you call `emailChecker()`, which again returns the resultset, `$check_res`. Line 70 counts the number of records in the result set to determine whether the email address already exists in the table. If the number of rows is less than 1, the address is not in the list and it cannot be unsubscribed.

In this case, the response message is stored in lines 75–76. However, if the number of rows is not less than 1, the user is unsubscribed (the record deleted) and the response

is stored in lines 84–88. At that point, the script breaks out of the `if...else` construct, skips down to line 101, and proceeds to print the HTML.

Figures 19.2 through 19.5 show the various results of the script, depending on the actions selected and the status of email addresses in the database.

**FIGURE 19.2**  
Successful  
subscription.



**FIGURE 19.3**  
Subscription  
failure.



**FIGURE 19.4**  
Successful  
unsubscribe  
action.





**FIGURE 19.5**  
Unsuccessful  
unsubscribe  
action.

Next, you create the form and script that sends along mail to each of your subscribers.

## Developing the Mailing Mechanism

With the subscription mechanism in place, you can create a basic form interface for a script that takes the content of your form and sends it to every address in your subscribers table. This is another one of those all-in-one scripts, called `sendmyemail.php`, and it is shown in Listing 19.3.

Before attempting to use the script in this section, make sure that you have read the section in Chapter 11, “Working with Forms,” regarding the configuration in your `php.ini` file. The `php.ini` file is required to send mail.

### NOTE

### LISTING 19.3 Send Mail to Your List of Subscribers

```

1: <?php
2: include 'ch19_include.php';
3: if (!$_POST) {
4:     //haven't seen the form, so display it
5:     $display_block = <<<END_OF_BLOCK
6:     <form method="POST" action="$_SERVER[PHP_SELF]">
7:
8:     <p><label for="subject">Subject:</label><br/>
9:     <input type="text" id="subject" name="subject" size="40" /></p>
10:
11:     <p><label for="message">Mail Body:</label><br/>
12:     <textarea id="message" name="message" cols="50"
13:         rows="10"></textarea></p>
14:     <button type="submit" name="submit" value="submit">Submit</button>
15:     </form>
16:     END_OF_BLOCK;
17: } else if ($_POST) {

```

**LISTING 19.3** Continued

---

```

17: //want to send form, so check for required fields
18: if (($_POST['subject'] == "") || ($_POST['message'] == "")) {
19:     header("Location: sendmymail.php");
20:     exit;
21: }
22:
23: //connect to database
24: doDB();
25:
26: if (mysqli_connect_errno()) {
27:     //if connection fails, stop script execution
28:     printf("Connect failed: %s\n", mysqli_connect_error());
29:     exit();
30: } else {
31:     //otherwise, get emails from subscribers list
32:     $sql = "SELECT email FROM subscribers";
33:     $result = mysqli_query($mysqli, $sql)
34:         or die(mysqli_error($mysqli));
35:
36:     //create a From: mailheader
37:     $mailheaders = "From: Your Mailing List
38:         <you@yourdomain.com>";
39:     //loop through results and send mail
40:     while ($row = mysqli_fetch_array($result)) {
41:         set_time_limit(0);
42:         $email = $row['email'];
43:         mail("$email", stripslashes($_POST['subject']),
44:             stripslashes($_POST['message']), $mailheaders);
45:         $display_block .= "newsletter sent to: ".$email."<br/>";
46:     }
47:     mysqli_free_result($result);
48:     mysqli_close($mysqli);
49: }
50: }
51: ?>
52: <!DOCTYPE html>
53: <html>
54: <head>
55: <title>Sending a Newsletter</title>
56: </head>
57: <body>
58: <h1>Send a Newsletter</h1>
59: <?php echo $display_block; ?>
60: </body>
61: </html>

```


---

As in Listing 19.2, the file of user-defined functions is included on line 2. Although only the database connection function is used in this file, there's no harm in having the other function in the file, as well.

The main logic of the script starts at line 3, where you determine whether the user has seen the form yet. If the presence of the `$_POST` variable is false, you know the user has not submitted the form; therefore, you must show the form.

Lines 5–15 create the form for sending the newsletter to your subscriber list, which uses `$_SERVER[PHP_SELF]` as the action (line 6), creates a text field called `subject` for the subject of the mail, and creates a `textarea` called `message` for the body of the mail to be sent.

At this point, the script breaks out of the `if . . . else` construct, and the HTML is printed. The form displays as shown in Figure 19.6.

A screenshot of a web browser window. The title bar says "Sending a Newsletter". The address bar shows "localhost/19/sendmyemail.php". The main content area has a heading "Send a Newsletter". Below the heading is a "Subject:" label followed by a text input field. Below that is a "Mail Body:" label followed by a large text area. At the bottom left of the form is a "Submit" button.

**FIGURE 19.6**  
Form for sending the bulk mail.

If the presence of `$_POST` is not `false`, the script should send the form to the email addresses in the `subscribers` table. Before sending the message, you must check for the two required items from the form in lines 18–20: `$_POST[ 'subject' ]` and `$_POST[ 'message' ]`. If either of these items is not present, redirect the user to the form again.

If the required items are present, the script moves on to line 24, which calls the database connection function. A query is issued in line 33, which grabs all the email addresses from the `subscribers` table. There is no order to these results, although you could throw an `order` by clause in there if you want to send them out in alphabetic order for whatever reason.

Lines 37–38 create a `From:` mail header, which is used inside the upcoming `while` loop, when the mail is sent. This header ensures that the mail looks like it is from a person and not a machine because you’ve specifically provided a value in this string. The `while` loop, which begins on line 40, extracts the email addresses from the resultset one at a time. On line 41, you use the `set_time_limit()` function to set the time limit to `0`, or “no limit.” Doing so allows the script to run for as long as it needs to.



**CAUTION**

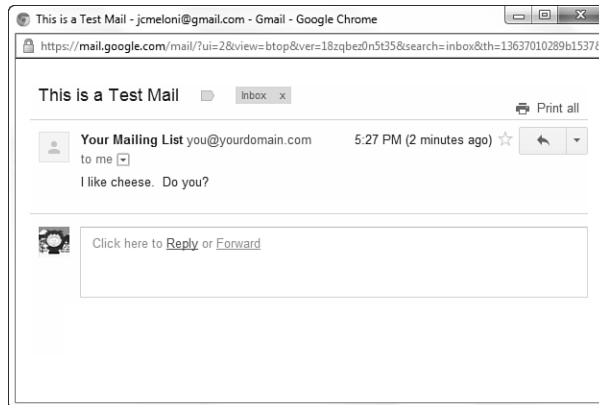
Because the script in Listing 19.3 simply executes the `mail()` function numerous times, it does not take into account the queuing factors in actual mailing list software, which are designed to ease the burden on your outgoing mail server. Using `set_time_limit()` does not ease its burden; it just allows the script to continue to run when it might have timed out before.

In lines 43–44, the mail is sent using the `mail()` function, inserting the values from the form where appropriate. Line 45 adds to a string that is later printed to the screen, which shows to whom the mail was sent. Figures 19.7 and 19.8 show the outcome of the script.

**FIGURE 19.7**  
Mail has been sent!



**FIGURE 19.8**  
The mail arrived safely.



## Summary

In this chapter, you applied your basic PHP and MySQL knowledge to the creation of a personal mailing list. Included were the database table creation, the subscribe and unsubscribe mechanisms, and the form and script for sending the mail.

## Q&A

**Q.** *How can I ease the burden on my mail server?*

**A.** Besides looking into packaged mailing list software, you can bypass the `mail()` function and talk directly to your SMTP server via a socket connection. Such an example is shown in the PHP manual for the `fsockopen()` function (<http://www.php.net/fsockopen>), as well as in other developer resource sites.

**Q.** *Where do bounced messages go?*

**A.** As with any email (not just those sent in the manner described in this chapter), bounces go to whatever address you specify in your `From:` or `Reply-to:` mail headers.

## Workshop

The workshop is designed to help you review what you've learned and begin putting your knowledge into practice.

## Quiz

1. Which PHP function sends mail?
2. Why is `$mysqli` named as a global variable in Listing 19.1?
3. What PHP function call causes the script to execute for as long as it needs to run?

## Answers

1. This is not a trick question. It's the `mail()` function!
2. Because the variable `$mysqli` is created and assigned a value in a function that is included in one script for use by another, the variable must be declared as global to ensure it is usable outside of the confines of the function in which it was created.
3. `set_time_limit(0)`

## Activities

1. Modify the `manage.php` script to display the user's email as part of the response message for any action that is taken.
2. Modify the `sendmyemail.php` script to add additional form fields that will correspond to section headings in the message string itself. Remember that when the form is submitted, those strings will have to be concatenated into one message string that is sent to the `mail()` function.

# Index

## Numbers

3D pie charts, creating, 269-270

## Symbols

+ (addition operators), 85-87

&& (and operator), 91-92

[ ] (array operators), 140

= (assignment operators), 77, 86-89

\* (asterisks), 305

comments, 72

wildcards, 32

\ (backslashes)

directives, 45

escaping string quotation marks, 113

time stamp conversions, 184

. (concatenation operators), 87-88

/ (division operators), 87

\$ (dollar signs), variable names, 76

... (ellipses)

color fills, 266

drawing, 264

= (equal to operator), 308

== (equivalence operators), 90

// (forward slashes), comments, 72

> (greater than operators), 91, 308

>= (greater than or equal to operators), 91, 308

=== (identical operators), 91

< (less than operators), 91, 308

<= (less than or equal to operators), 91, 308

% (modulus operators), 87

\* (multiplication operators), 87

! (not operators), 92

!= (nonequivalence operators), 90

!= (not equal to operator), 308

|| (or operators), 91-92

() (parentheses), subqueries, 315

% (percent signs)

conversion specifications, 160-162

log-formatting directives, 510

wildcards, 32

# (pound signs)

comments, 72

directives, 45

PHP/Apache integration, 62

"" (quotation marks)

escaping strings, 113

MySQL field names, 324

;(semicolons)

instruction terminators, 76

statements, 70

- (subtraction operators), 87

? (ternary operator), 105

## A

%a format string option (DATE\_FORMAT() function), 337

%A log-formatting directive, 510

ab (ApacheBench), 559-561

aborted\_connects status variable, 602

abs() function, 121

abuse prevention, 563-564

accept mechanisms, 559

## access

authentication

authoritative information, 494

back-end storage, 494

browsers, 493

client, 492-493

combining with access control rules, 500

database file-based, 496-497

defined, 491

denying access, 494

directives, 493-494

etc/passwd file, 507

file-based, 495-496

realms, 493

user management, 494

authorization, 492

class properties with object methods, 153

constants, 94

cookies, 214-215

log files, 50

global variables, 127

reports, creating, 521-522

restricting based on

cookies, 502-506

HTTP methods, 501

rules

clients, 498

combining with authentication, 500

domain names, 498

environment variables, 498

evaluating, 499-500

implementing, 497-498

- IP addresses, 497
  - security, 500
- session variables, 219-223
- variables, 126-128
- XML in PHP
  - DOM functions, 544-546
  - SimpleXML functions, 546-549
- Access denied error messages, 26**
- AccessFileName directive, 49**
- ACTION argument, 190**
- action-based permissions, 30**
- AddCharset directive, 530**
- addentry.php script, 392-396, 409-413**
- addFive() function, 132**
- addition (+) operators, 85-87**
- AddLanguage directive, 530**
- addNums() function, 124**
- address books**
  - birthdays, adding, 414
  - database tables, creating, 387-390
  - email tables, 389
  - fax tables, 389
  - field names, 388
  - master name tables, 388-389
  - personal notes tables, 390
  - telephone tables, 389
- include files, creating, 390-391
- menus, creating, 391
- records
  - adding, 392-398
  - deleting, 404-406
  - subentries, adding, 406-413
  - viewing, 398-404
- addresses**
  - IP
    - access control, 497
    - reverse DNS lookups, 512
  - listening, 54
- addtocart.php script, 456-458**
- alert option (LogLevel directive), 516**
- algorithms**
  - digest, 493, 576
  - symmetric key cryptography, 574
- ALL privilege (MySQL), 31**
- Allow,Deny argument (Order directive), 499**
- Allow directive (access rules), 497**
- AllowOverride directive, 50**
- ALTER privilege (MySQL), 31**
- alternative calendars, 489**
- ampersands (&&), 91-92**
- analyzing logs, 518**
- and operator, 92**
- Apache**
  - access control
    - clients, 498
    - combining with authentication, 500
    - domain names, 498
    - environment variables, 498
    - evaluating rules, 499-500
    - implementing rules, 497-498
    - IP addresses, 497
    - restricting based on cookies, 502-506
    - restricting based on HTTP methods, 501
  - authentication
    - authoritative information, 494
    - back-end storage, 494
    - browsers, 493
    - client, 492-493
    - combining with access control rules, 500
    - database file-based, 496-497
    - denying access, 494
    - directives, 493-494
    - etc/passwd file, 507
    - file-based, 495-496
    - realms, 493
    - user management, 494
- Caching Guide website, 562
- changelog example, 606
- clean build, 56
- commands, 51-53
- configuring, 45
  - conditional containers, 48-49
  - containers, 46-47
  - directives, 45-46
  - per-directory configuration files, 49-50, 57
  - ServerRoot directive, 49
- container, 48
- content-based content negotiation, 530
- installation
  - binary, 39
  - Linux/UNIX, 39-42
  - Mac, 42
  - methods, selecting, 38
  - source code, 38
  - Windows, 42-44
- internationalization configuration changes, 530
- licenses, 43
- logging, 509
  - analysis, 518
  - errors, 515-519
  - files, 50-51, 514
  - formatting, 510-513
  - hostname lookups, 512
  - hostname resolution, 517
  - identity checks, 513
  - images, 514, 524
  - programs, 515
  - request logs, creating, 509
  - rotation, 518
  - status codes, 513
- mod ssl module, 580-582
- modifying without upgrading, 608
- MPM settings, 556
- News and Announcements list website, 605

- performance
  - abuse prevention, 563-564
  - caching, 562
  - file system settings, 558-559
  - load distribution, 562
  - load testing, 559-561
  - mapping files to memory, 561
  - network settings, 559, 563
  - operating system limits, 556-557
  - speed, 570
  - status settings, 559
  - transmitted data, reducing, 562
- PHP installation on Linux/UNIX, 60-62
- PHP integration, 62-64
- secure mode, starting, 585
- starting, 53-54
- troubleshooting
  - access denied, 56
  - binding to ports
    - permissions, 55
  - existing web servers, 55
  - group settings, 56
- upgrading, 608-609
- versions, 37-38
- virtual hosting, 564
  - DNS, 564
  - IP-based, 564-565, 570
  - mass, 568-569
  - name-based, 564-570
- website, 37-39
- ApacheBench (ab), 559-561**
- apachectl utility, 53**
- APIs**
  - output, 552
  - ProgrammableWeb website, 553
- appending files, 239, 245-246**
- application frameworks**
  - benefits, 611
  - CakePHP, 616
  - choosing, 614
  - CodeIgniter, 616-617
  - content management systems, 612
  - defined, 611
  - evaluating, 614
  - MVC pattern, 612
    - components, 613
    - displaying, 612
    - flow, 613
    - websites, 614
  - Symfony, 614
  - Wikipedia listing of, 614
  - Yii, 614
  - Zend, 615
- application localization**
  - character sets, 528-529
  - environment modifications, 530-531
  - flags for language selections, 535
  - gettext() function, 536-537
  - internationalization, 527
  - locales, 528
  - numbers/dates/currency, 538
  - page structures, 531
    - language definition file, 531-532
    - language selector, 535
    - locale selection formats, 536
    - string definition file, 533
    - welcome script, 534
- applying directives, 47**
- APR (Apache) website, 41**
- arcs**
  - color fills, 266
  - drawing, 264
- arguments**
  - ACTION, 190
  - Allow,Deny, 499
  - AllowOverride directive, 50
  - CustomLog directive, 514
  - default values, setting, 130-132
  - defined, 120
  - Deny,Allow, 499
  - directives, 45
  - ENCTYPE, 207
  - flock() function, 247
  - HostNameLookups directive, 512
  - LogFormat directive, 512
  - Mutual-Failure, 500
  - optional, setting, 131
  - rotatelog/rotatelog.exe utilities, 518
  - swapping, 167-168
  - syslog, 516
  - TYPE, 207
  - variable references, passing, 132-133
- arithmetic operators, 86-87**
- array() function, 81, 140**
- array operator ([ ]), 140**
- arrays, 79**
  - associative
    - creating, 141-142
    - getdate() function, 180
  - contact forms, 146
  - creating, 140-141
  - defined, 139
  - functions
    - array\_keys(), 145
    - array\_merge(), 145
    - array\_pop(), 145
    - array\_push(), 145
    - array\_shift(), 145
    - array\_unshift(), 145
    - array\_values(), 145
    - count(), 144
    - each(), 144
    - foreach(), 144
    - list(), 144
    - reset(), 145
    - shuffle(), 145
    - sizeof(), 144

- HTML form input, accessing, 191-194
  - keys, 140
  - multidimensional
    - creating, 142-144
    - dimensions, 146
  - session variables, adding, 221
  - strings, breaking, 179
  - values, 140
  - website, 146
  - `array_keys()` function, 145
  - `array_merge()` function, 145
  - `array_pop()` function, 145
  - `array_push()` function, 145
  - `array_shift()` function, 145
  - `array_unshift()` function, 145
  - `array_values()` function, 145
  - ASP tags, 69
  - assignment operators (=), 77, 86-89
  - associative arrays
    - creating, 141-142
    - `getdate()` function, 180
  - asterisks (\*), 305
    - comments, 72
    - wildcards, 32
  - asymmetric cryptography, 574-576
  - auth cookies, testing, 506
  - `AuthDBMGroupFile` directive, 496
  - `AuthDBMUserFile` directive, 496
  - authentication, 573
    - authoritative information, 494
    - back-end storage, 494
    - browsers, 493
    - client, 492-493
    - combining with access control rules, 500
    - database file-based, 496-497
    - defined, 491
    - denying access, 494
    - directives, 493-494
    - etc/passwd file, 507
    - file-based, 495-496
    - MySQL privileges, 29-31
    - realms, 493
    - SSL, 576-578
    - user management, 494
  - `AuthGroupFile` directive, 495
  - `AuthName` directive, 493
  - authorization, 492
  - authorized users tables, creating, 502-503
  - Authorize.Net, 463
  - `AuthType` directive, 493
  - `AuthUserFile` directive, 495
  - availability
    - functions, 133-134
    - variables, 77
  - awstats, 519
- ## B
- `%b` format string option (`DATE_FORMAT()` function), 337
  - `%b` log-formatting directive, 511
  - back-end storage, 494-496
  - background colors, 263
  - backslashes (\)
    - directives, 45
    - escaping string quotation marks, 113
    - timestamp conversions, 184
  - basic authentication, 492
  - `BEGIN` command, 352
  - `benchmark()` function, 590-591
  - benchmarking, 590-591
  - `BETWEEN` operator, 309
  - `BIGINT` data type, 298
  - binaries
    - Apache installation, 39
    - distributions, 17
    - outputting, 256
    - server binary commands, 51-53
  - `BINARY` keyword, 310
  - `bindtextdomain()` function, 537
  - `BLOB` data type, 300
  - `books.xml` document, 542
  - booleans, 79
  - Boyce-Codd normal forms, 294
  - breadcrumb trails, 447
  - break statements, 109-111
  - browsers
    - Apache, starting, 54
    - authentication, 493
    - padlock icons, 577
  - built-in functions, 120
  - bulk mail, sending, 383
- ## C
- c command-line option, `htpasswd` utility, 496
  - `%c` format string option (`DATE_FORMAT()` function), 337
  - `%C` log-formatting directive, 511
  - `CacheFile` directive, 561
  - CakePHP, 616
  - `calendar_events` table, 474
  - calendars
    - alternative, 489
    - events, adding, 474-482
    - HTML forms, building, 469-470
    - libraries, creating, 483-488
    - tables, creating, 471-474
    - user input, 467-468
  - `CALL` command, 355
  - calling functions, 120-121, 135
  - Can't connect to server error message, 26
  - canvas area (images), 263
  - `CAPTCHAs`, 279
  - CAs (certification authorities), 577
  - case sensitivity
    - comparison operators, 310
    - constants, 94
    - functions, 123
    - XML documents, 543
  - casting variables, 82-84

categories (online storefront items), displaying, 441-444

cat\_desc field (store\_categories table), 438

cat\_id field (store\_items table), 438

cat\_title field (store\_categories table), 438

certificates (digital), 577

- chaining, 577
- key pairs, creating, 582-583
- self-signed, creating, 584
- signing, 577
- signing requests, 583-584
- testing, 577
- X.509, 577-578

certification authorities (CAs), 577

CGI errors, logging, 515

CHAR\_LENGTH() function, 323

CHARACTER\_LENGTH() function, 323

characters

- files, reading, 243-244
- names, 345
- sets, 528
  - multibyte, 528
  - MySQL installation, 24
  - single-byte, 528
  - unrecognizable characters, 529

CHAR(M) data type, 300

CHARSET variable, 532

charts (pie), 267-270

- 3D, 269-270
- dynamic data, 281
- slices, 268

\$check\_res variable, 379

checkdate() function, 185, 468

checkout actions (shopping carts), 463-464

checkout forms (shopping carts), 463

children (XML documents), 542

ciphertext, 574

classes

- date\_pulldown, 483, 487-488
- defined, 150
- inheritance, 155
- properties
  - accessing with object methods, 153
  - values, modifying, 154

clauses

- LIMIT, 307
- ON, 313
- ORDER BY
  - date/time functions, 333
  - DELETE command, 321
  - SELECT command, 305
- WHERE, 308
  - BETWEEN operator, 309
  - comparison operators, 308
  - LIKE operator, 309
  - logical operators, 309

clean builds (Apache), 56

cleaning up strings

- ltrim() function, 174
- rtrim() function, 173
- strip\_tags() function, 174
- trim() function, 173

CLF (Common Log Format), 512

clients

- access control, 498
- authentication, 492-493
- requests, tracking, 50

closing files, 239

code blocks

- brackets, 116
- echo statements, 114
- HTML mode, returning, 115

code snippets, 520

CodeIgniter, 616-617

collision resistant message digests, 576

colors (images)

- allocating, 263

- background, 263
- fills, 266
- RGB values, 262

column command, 252-253

columns\_priv tables, 29

combined assignment operators, 88-89

commands

- Apache, 51-53
- BEGIN, 352
- CALL, 355
- certificate signing requests, 583
- column, 252-253
- COMMIT, 350-352
- CREATE TABLE, 301-302
- DELETE, 320-321
  - conditional, 321-322
  - MySQL privileges, 33
  - ORDER BY clause, 321
  - subqueries, 315
- EXPLAIN, 594
- FLUSH, 595-596
- FLUSH HOSTS, 596
- FLUSH LOGS, 596
- FLUSH PRIVILEGES, 33, 595
- FLUSH TABLES, 596
- GRANT, 31-32
- INNER JOIN, 312
- INSERT, 303-304
  - authorized user tables, creating, 502
  - MySQL users, adding, 31
  - syntax, 302
- JOIN, 312
- key pairs, creating, 582
- kill, 52
- LEFT JOIN, 313-314
- ln, 558
- make
  - Apache, building, 41
  - PHP installation, 61
- make install, 41, 61



- OPTIMIZE, 603
- OPTIMIZE TABLE SQL, 593
- ps, 27
- REPLACE, 319-320
- REVOKE, 33
- RIGHT JOIN, 314
- ROLLBACK, 350-352
- running
  - exec() function, 254-255
  - passthru() function, 256
  - system() function, 255
- SELECT, 304-305
  - \* symbol, 305
  - limiting results, 307
  - ordering results, 305-306
  - subqueries, 315
  - syntax, 304
- SHOW, 596-597
- SHOW COLUMNS, 599
- SHOW CREATE TABLE, 599
- SHOW DATABASES, 598-599
- SHOW GRANTS, 597
- SHOW INDEX, 600
- SHOW STATUS, 593, 601-602
- SHOW TABLE STATUS, 600
- SHOW VARIABLES, 601-602
- ulimit, 556
- UPDATE
  - conditional, 317
  - existing values, 318-319
  - subqueries, 315
  - tables, 316-317
- who, 252
- comments, 72-73**
- COMMIT command, 350-352**
- Common Log Format (CLF), 512**
- comparison operators, 90-91**
  - case sensitivity, 310
  - WHERE clauses, 308
- complementary keys, 575**
- component libraries, 615**
- CONCAT() function, 323**
- CONCAT\_WS() function, 324**
- concatenation functions, 322-325**
- concatenation operators (.), 87-88**
- concurrent connections, 24**
- conditional containers, 48-49**
- conditional DELETE command, 321-322**
- conditional statements (included files), 232**
- conditional updates (tables), 317**
- confidentiality, 573**
  - public key cryptography, 575-576
  - symmetric cryptography, 574
- configuration files, customizing, 52**
- configuring**
  - Apache
    - conditional containers, 48-49
    - containers, 46-48
    - directives, 45-46
    - files, 53-54
    - installation software, 40-41
    - per-directory configuration files, 49-50, 57
    - script, 40-41
    - ServerRoot directive, 49
  - cookies, 215
  - mail() function, 200-201
  - PHP, 60-62
  - php.ini file, 65
  - SSL, 585
- connect\_timeout variable, 602**
- connections**
  - MySQL, 28
  - MySQL with PHP
    - closing, 359
    - creating, 358
    - errors, 359-361
    - queries, executing, 360-361
    - syntax, 358
  - SSL, 578
  - status variable, 602
- constants, 94-95**
- constructors**
  - defined, 483
  - objects, 154
- contact forms, 146**
- containers, 46**
  - conditional, 48-49
  - defined, 45
  - directories, 47
  - files, 47
  - Limit, 501
  - LimitExcept, 501
  - syntax, 47
  - URLs, 47
  - virtual servers, 47-48
  - VirtualHost, 565
- content**
  - management systems, 612
  - negotiation, 558
  - structure (XML documents), 542
- continue statements, 111-112**
- control information (certificates), 577**
- control scripts (Apache), 53**
- conversion functions, 342-343**
- conversion specifiers**
  - printf() function, 160-162
  - string field width, 165-166
- converting**
  - string text case, 176-177
  - timestamps to dates
    - date() function, 182-184
    - getdate() function, 180-182
- \$\_COOKIE superglobal, 77**
- cookies**
  - access restrictions, 502
  - auth cookie, testing, 506
  - authorized users tables, creating, 502-503
  - login forms, creating, 503
  - login scripts, creating, 503-505

- accessing, 214-215
- auth, testing, 506
- \$\_COOKIE superglobal, 215
- defined, 213
- deleting, 217
- disabling, 226
- domain field, 214
- expiration dates, 214-216
- headers, 214
- HTTP\_COOKIE environment variable, 215
- path field, 214
- printing, 215
- security, 507
- setting, 215
- size, 213
- viewing, 216
- count() function, 144**
- \$count variable, 473**
- CPUs (MySQL), 589, 603**
- CREATE privilege (MySQL), 31**
- CREATE TABLE command, 301-302**
- crit option (LogLevel directive), 516**
- cross-platform editors, 536**
- crowdsourcing services, 536**
- CURDATE() function, 341**
- currency, 538**
- CURRENT\_DATE() function, 341**
- current dates/times, retrieving, 180, 341-343**
- CURRENT\_TIME() function, 341**
- CURRENT\_TIMESTAMP() function, 342**
- CURTIME() function, 341**
- customizing. See modifying**
- CustomLog directive, 514**

## D

- %D format string option (DATE\_FORMAT() function), 337**
- %D log-formatting directive, 510**

## D option (httpd/httpd.exe commands), 52

- data**
  - binaries
    - Apache installation, 39
    - distributions, 17
    - outputting, 256
    - server binary commands, 51-53
  - integrity (security), 573-576
  - JSON, 549
    - API output, 552
    - formatting, 550-552
    - Google search output, 552
    - loading/displaying, 551
    - output, creating, 553
  - MySQL
    - inserting with PHP, 363-367
    - retrieving with PHP, 367-369
    - SQL injections, avoiding, 362-363
  - passing to external applications, 252-253
  - types, 78
    - changing, 82-84
    - date/time, 299-300
    - defining, 298
    - numeric, 298-299
    - standard, 78
    - string, 300-301
    - testing, 79-80, 85
  - XML storage, 553

## databases

- address books, creating, 387-390
- email tables, 389
- fax tables, 389
- field names, 388
- master name tables, 388-389
- personal notes tables, 390
- telephone tables, 389

- custom logs, creating
  - code snippets, creating, 520-521
  - database tables, creating, 519
  - sample reports, 521-523
- dates/times, storing, 187
- designing
  - good, 283-284
  - process, 292-293
- discussion forum tables, creating, 417-418
- file-based authentication, 496-497
- information, retrieving, 598-599
- InnoDB storage engine, 350
- maintenance, 284
- normalization, 283, 289
  - additional forms, 294
  - first normal forms, 290
  - flat tables, 289-290
  - normal forms, 289
  - redundancy, 290
  - second normal forms, 291
  - third normal forms, 291-292
- performance, 283
- shopping cart tables
  - adding items to cart, 456-458
  - cart, viewing, 458-461
  - checkout actions, 463-464
  - checkout forms, creating, 463
  - date items were added to cart field, 452
  - fields, 451-453
  - id fields, 452
  - item inventory, 465
  - real-time credit card processing, 453
  - removing items from cart, 461-462
  - selections, holding, 452
  - shipping addresses, 453

- storefront integration, 453-456
- users, identifying, 452
- stored procedures
  - benefits, 353
  - calling, 355
  - creating, 354
  - defined, 353
  - syntax, 354
  - website, 355
- storefront tables
  - categories of items, displaying, 441-444
  - creating, 437-439
  - field names, 438
  - items, displaying, 445-447
  - records, adding, 439-441
  - store\_categories table, 438
  - store\_item\_color table, 439
  - store\_item\_size table, 439
  - store\_items table, 438
- table relationships
  - many-to-many, 287-288
  - one-to-many, 286
  - one-to-one, 285
  - types, 284
- transactions
  - BEGIN command, 352
  - COMMIT command, 350-352
  - defined, 349
  - displaying versus inserting data, 355
  - online storefront example, 351-353
  - ROLLBACK command, 350-352
  - syntax, 350-351
  - website, 351
  - users, adding/deleting, 497
- DATE\_ADD() function, 339-341**
- date\_added** field (shopping cart database tables), 452
- DATE data type, 299**
- date\_format() function, 343**
  - discussion forum topic lists, 425
  - options, 337
  - syntax, 337
- date() function, 182-184**
  - files, 236
  - localization, 538
- date\_pulldown** class, 483, 487-488
- date\_pulldown** library, 483
- DATE\_SUB() function, 339-341**
- dates/times**
  - calendars
    - events, adding, 474-482
    - HTML forms, building, 469-470
    - libraries, creating, 483-488
    - tables, creating, 471-474
    - user input, 467-468
  - current, retrieving, 180
  - data types, 299-300
  - databases, 187
  - dates, testing, 185
  - files, 236
  - functions, 331
    - arithmetic, 339-341
    - conversion, 342-343
    - current, 341-343
    - days, 331-333
    - formatting, 337-339
    - hours, 336
    - minutes, 336
    - months, 333
    - seconds, 336
    - weeks, 334-336
    - years, 334
  - HH:MM:SS time format, 341
  - localization, 538
  - timestamps
    - converting to dates, 180-184
    - creating, 184-185
- defined, 180
- UNIX epoch, 180
- website resources, 186
- YYYY-MM-DD format, 341
- DATETIME data type, 300**
- day functions, 331-333**
- day\_select() function, 486**
- DAYNAME() function, 333**
- DAYOFMONTH() function, 332**
- DAYOFWEEK() function, 331**
- DAYOFYEAR() function, 332**
- db tables, 29**
- Debian packages, 17**
- debug option (LogLevel directive), 517**
- DECIMAL(M,D) data type, 299**
- declaring**
  - functions, 122-123
  - objects, 150
  - properties, 157
  - variables, 76, 125-126
- decrementing integers, 89-90**
- decryption, 574**
- define() function, 94-95**
- defineStrings() function, 533-534**
- defining functions, 121**
- delentry.php** script, 404-406
- DELETE command, 320-321**
  - conditional, 321-322
  - MySQL privileges, 33
  - ORDER BY clause, 321
  - subqueries, 315
- DELETE privilege (MySQL), 31**
- deleting**
  - address book records, 404-406
  - cookies, 217
  - database users, 497
  - directories, 249
  - files, 238
  - MySQL privileges, 33-34
  - session variables, 224

- shopping cart items, 461-462
- table records, 320-322
- tags from strings, 174
- whitespace from strings, 173
- Denial of Service (DoS), 563**
- Deny directive (access rules), 497**
- Deny,Allow argument (Order directive), 499**
- DES algorithm, 574**
- designing databases**
  - good, 283-284
  - process, 292-293
- destroying sessions, 223-224**
- die() function, 239**
- digest algorithms, 493, 576**
- digest authentication, 492**
- digital certificates, 577**
  - chaining, 577
  - key pairs, creating, 582-583
  - self-signed, creating, 584
  - signing, 577
  - signing requests, 583-584
  - testing, 577
  - X.509, 577-578
- directives**
  - \ (backslashes), 45
  - # (pound sign), 45
  - AccessFileName, 49
  - AddCharset, 530
  - AddLanguage, 530
  - Allow, 497
  - AllowOverride, 50
  - arguments, 45
  - AuthDBMGroupFile, 496
  - AuthDBMUserFile, 496
  - authentication, 493-494
  - AuthGroupFile, 495
  - AuthName, 493
  - AuthType, 493
  - AuthUserFile, 495
  - CacheFile, 561
  - containers, 46-49
  - CustomLog, 514
  - defined, 45
  - Deny, 497
  - documentation, 46
  - ErrorLog, 515
  - flag, 65
  - HostNameLookups, 512
  - IdentityCheck, 513
  - include\_path, 233
  - KeepAliveTimeout, 563
  - Listen, 54, 565
  - LoadModule, 585
  - LogFormat, 512-514
  - log-formatting, 510-511
  - LogLevel, 516-517
  - MMapFile, 561
  - NameVirtualHost, 566
  - Options
    - mass virtual hosting, 569
    - parameters, 558
  - Order, 499-500
  - overriding, 50
  - php.ini file, 65
  - Require, 494
  - Satisfy, 500
  - ScoreBoardFile, 559
  - ScriptAlias, 569
  - ServerAlias, 567
  - ServerName
    - configuration files, checking, 54
    - validity, 57
  - ServerRoot, 49
  - SSLCertificateFile, 585
  - SSLCertificateKeyfile, 585
  - SSLEngine, 585
  - syntax, 45-46
  - TimeOut, 563
  - TransferLog, 514
  - value, 65
  - VirtualDocumentRoot, 568
  - VirtualDocumentRootIP, 569
  - VirtualScriptAlias, 569
  - VirtualScriptAliasIP, 569
  - website, 46
- directories**
  - creating, 248-249
  - deleting, 249
  - directives, applying, 47
  - file/directory conformation, validating, 234
  - listing (UNIX), 254
  - opening, 249
  - per-directory configuration files, 49-50, 57
  - PHP, 64
  - reading, 249-251
  - source files, 60
  - subdirectories, 592
  - usr/local/apache2, 41
  - usr/local/php/lib, 65
  - usr/local/src, 60
  - usr/src, 60
- <Directory> container, 47**
- <DirectoryMatch> container, 47**
- disabling**
  - cookies, 226
  - per-directory configuration files, 50
- discussion forums**
  - database tables, creating, 417-418
  - included files, 418-419
  - multiple, creating, 434
  - posts
    - adding, 430-433
    - displaying, 426-429
    - first entry, creating, 420-422
  - topics
    - lists, displaying, 423-426
    - form, creating, 419
    - script, creating, 420-422

**\$\_display\_block strings, 399****display scripts (localized), 534****displaying**

- address book records, 398-404
- cookies, 216
- discussion forum topic lists, 423-426
- form URL values, 210
- JSON data, 551
- multiple spaces (HTML), 163
- MVC pattern, 612
- object properties, 152
- online storefront
  - categories of items, 441-444
  - items, 445-447
- posts (discussion forums), 426-429
- shopping cart items, 458-461

**distribution files, 41, 63****division operators (/), 87****DNS**

- reverse lookups, 512
- round-robin, 565
- virtual hosting, 564

**do while loops, 107-108****Document Object Model (DOM), 544****documents**

- formatting as text, 163
- included files, 229-231
  - calling twice, 233
  - conditional statements, 232
  - loops, 232
  - performance, 257
  - portability, 233
  - return values, 231
- XML, 541-543
  - capabilities, 543
  - case sensitivity, 543
  - children, 542
  - content structure, 542
  - parsing with DOM functions, 544-546

- parsing with SimpleXML functions, 546-549

- prologs, 541
- root elements, 542
- sample, 542
- tags, 543
- XML specification, 542

**doDB() function, 375, 379****dollar signs (\$), variable names, 76****DOM (Document Object Model), 544-546****domain fields (cookies), 214****domain names, 498****domexample.php, 546****DoS (Denial of Service), 563****double argument (HostNameLookups directive), 512****double data types, 79****DOUBLE(M,D) data type, 299****downloading**

- Apache source code, 39
- CakePHP, 616
- CodeIgniter, 617
- MySQL, 16
- PHP distribution files, 63
- XAMPP installation, 8
- Zend Framework, 615

**drawing images**

- background colors, 263
- canvas area, 263
- colors, 263
- custom fonts, 279-280
- from existing images, 271-273
- ImageColorAllocate() function, 263
- ImageCreate() function, 263
- lines, 264-265
- pie charts, 267-270, 281
- scripts, 278-280
- shapes, 264-265
- transparent, 272-273
- user input, 273-277

- x-axis coordinates, 264

- y-axis coordinates, 264

**DROP privilege (MySQL), 31****Drupal website, 612****E****%e format string option (DATE\_FORMAT() function), 337****%e log-formatting directive, 510****each() function, 144****echo() function, 120****echo statements, 114**

- multidimensional arrays, 144
- PHP scripts, 70

**ellipses (...)**

- color fills, 266
- drawing, 264

**else clauses, 100-101****elseif clauses, 101-102****email**

- feedback forms
  - creating, 201-202
  - formatting, 205
  - sending, 202-203
- fields (subscribers tables), 374
- HTML formatting, 205
- sending, 200-201
- tables (online address books), 389

**emailChecker() function, 375, 379****emerg option (LogLevel directive), 516****encryption**

- keys, 574
- passwords, 495
- public key cryptography, 575-576
- symmetric cryptography, 574

**ENCTYPE argument (file upload forms), 207****ending**

- sessions, 223-224
- statements, 68-70
- tags, 68-70, 73

- ENUM data type, 301
- `$_ENV` superglobal, 78
- environment modifications (internationalization)
  - Apache configuration changes, 530
  - MySQL configuration changes, 531
  - PHP configuration changes, 530
- environment variables
  - access control, 498
  - HTTP\_COOKIE, 215
  - PATH, 64
- equal signs (=)
  - assignment operators, 77, 86
  - equivalence operators), 90
  - identical operators (===), 91
- equal to operator (=), 308
- equivalence operators (==), 90
- error option (LogLevel directive), 517
- ErrorLog directive, 515
- errors
  - logging
    - Apache, 51
    - files, 515
    - importance levels, 516-517
    - monitoring, 519
    - programs, 516
    - UNIX syslog daemon, 516
  - messages
    - Access denied, 26
    - Can't connect to server, 26
    - MySQL privilege authentication, 29
    - MySQL/PHP
      - connections, 361
- escapeshellarg() function, 256
- escapeshellcmd() function, 256
- escaping
  - string quotation marks, 113
  - user input elements, 256

- etc/passwd files, 507
- evaluating
  - access control rules, 499-500
  - application frameworks, 614
- events (calendar), adding, 474-482
- exclamation points (!), 92
- exec() function, 254-255
- executability, 235
- exit statements, 200
- expiration dates (cookies), 214-216
- EXPLAIN command, 594
- explode() function, 179
- expressions, 85-86
- Extensible Markup Language.
  - See XML
- extensions (PHP), 609
- external applications, passing data, 252-253
- external processes, 557
- extracting portions of strings, 171

## F

- %f log-formatting directive, 511
- f option (httpd/httpd.exe commands), 52
- fax tables (online address books), 389
- fcose() function, 239
- feedback forms
  - creating, 201-202
  - emailing, 202-203
  - formatting, 205
- feof() function, 240-241
- fgetc() function, 243-244
- fgets() function, 240-241
- field width specifiers (strings), 164-166
- fields (online storefront database tables), 438-439
- fifth normal forms, 294
- \$file\_array variable, 209
- file-based authentication, 495-496
- \$file\_dir variable, 209
- file\_exists() function, 234
- file\_get\_contents() function, 244-245, 551
- \$file\_name variable, 209
- file\_put\_contents() function, 246-247
- FILE privilege (MySQL), 31
- file system access settings, 558-559
- fileatime() function, 236
- filectime() function, 236
- filemtime() function, 236
- files
  - Apache configuration
    - conditional containers, 48-49
    - containers, 46-48
    - directives, 45-46
    - per-directory configuration files, 49-50, 57
    - ServerRoot directive, 49
  - Apache log, 50-51
  - appending, 239, 245-246
  - closing, 239
  - configuration, 52-54
  - content negotiation, 558
  - creating, 238
  - date/time information, 236
  - deleting, 238
  - descriptors, 556
  - directives, applying, 47
  - distribution, 41
  - error logs, 515
  - etc/passwd, 507
  - executability, 235
  - existence, checking, 234
  - file/directory confirmation, validating, 234
  - groups, 495
  - htaccess, 49
  - httpd.conf, 45
  - included, 229-231
    - calling twice, 233
    - conditional statements, 232

- discussion forums, creating, 418-419
- loops, 232
- mailing lists, 375
- online address books, creating, 390-391
- performance, 257
- portability, 233
- return values, 231
- INSTALL, 18
- language definition, 531-532
- locking, 247-248
- logging to, 514
- makefiles, 40
- mapping to memory, 561
- my.cnf, 592
- navigating, 242
- opening, 238, 258
- per-directory configuration, 49-50, 57, 558
- PHP distribution, 63
- php.ini, 65, 381
- phpinfo.php, 65
- reading, 235, 239
  - arbitrarily, 241-243
  - characters, 243-244
  - entire contents, 244-245
  - line by line, 240-241
  - popen() function, 251
- README, 18
- robots.txt, 564
- scoreboard, 559
- size, determining, 235-236
- source, directories, 60
- status, checking, 235
- string definition, 533
- testing, 236-238
- translation catalog, 536
- upload forms
  - \$\_FILES superglobal, 206
  - creating, 207
  - scripts, creating, 208-209
  - size, 211
  - upload names, 209
  - users, back-end storage, 495
  - writing, 235, 239
    - file\_put\_contents() function, 246-247
    - fopen() function, 245
    - fwrite() function, 246
- <Files> container, 47
- \$\_FILES superglobal, 78, 206
- filesize() function, 235-236
- <FilesMatch> container, 47
- fills (images), 266
- finding
  - error logs, 515
  - PHP text editors, 68
  - string lengths, 169
  - substrings, 170-171
- first normal forms, 290
- flag directives, 65
- flat tables, 289-290
- FLOAT(M,D) data type, 299
- floating data types, 79
- flock() function, 247-248
- flow control
  - code blocks 114-116
  - loops, 105
    - breaking, 109-111
    - do while, 107-108
    - for, 108-109
    - foreach, 143, 209
    - included files, 232
    - infinite, 108
    - iterations, 106
    - nesting, 112-113
    - skipping iterations, 111-112
    - while, 106-107, 252, 429
- MVC pattern, 613
- switching flow
  - if else statements, 100-101
  - if elseif statements, 101-102
  - if statements, 100
  - switch statements, 103-104
  - ternary operators, 105
- FLUSH command, 595-596
- FLUSH HOSTS command, 596
- FLUSH LOGS command, 596
- FLUSH PRIVILEGES command, 33, 595
- FLUSH TABLES command, 596
- FollowSymLinks parameter, 558
- fonts
  - custom, 279-280
  - images, 275
- fontWrap() function, 132
- fopen() function, 238, 245
- for statements, 108-111
- foreach() function, 144
- foreach loops, 143, 209
- foreign languages. *See* localization
- formatting
  - date/time functions, 337-339
  - documents, 163
  - email to HTML, 205
  - JSON data, 552
  - locale selections, 536
  - logging
    - CLF (Common Log Format), 512
    - defining, 512
    - directives, 510-511
    - host name lookups, 512
    - identity checks, 513
    - status codes, 513
  - strings
    - argument swapping, 167-168
    - field width specifiers, 164-166
    - printf() function, 160-164
    - storing, 168
- forms
  - checkout (shopping carts), 463
  - feedback
    - creating, 201-202

- emailing, 202-203
- formatting, 205
- file uploads, 206
- `$_FILES` superglobal, 206
  - creating, 207
  - scripts, creating, 208-209
  - size, 211
- hidden fields, 197-198
- HTML
  - accessing input via arrays, 191-194
  - calendar, building, 469-470
  - creating, 189-191
  - input, reading, 190
  - PHP combination, 194-196
- input, 419
- redirecting users, 198-200
- server headers, 198
- subscribe/unsubscribe, 376-381
- URL values, viewing, 210
- user login, 503
- forum\_posts table, 418**
- forum\_topics table, 418**
- forward slashes (//), 72**
- fourth normal forms, 294**
- fputs() function, 246**
- FQDN (fully qualified domain name), 565, 582**
- frameworks (application)**
  - benefits, 611
  - CakePHP, 616
  - choosing, 614
  - CodeIgniter, 616-617
  - content management systems, 612
  - defined, 611
  - evaluating, 614
  - MVC pattern, 612-614
  - Symfony, 614
  - Wikipedia listing of, 614
  - Yii, 614
  - Zend, 615

- fread() function, 241-243**
- From headers (email), 200, 203**
- FROM\_UNIXTIME() function, 342**
- fseek() function, 242-243**
- fully qualified domain name (FQDN), 565, 582**
- function\_exists() function, 133**
- function statement, 121**
- functions. See individual function names**
- fwrite() function, 246**

## G

- GD graphics library, 262**
- Get Localization, 536**
- GET method (forms), 191**
- `$_GET` superglobal, 77**
- getdate() function, 180-182, 468**
- gettext() function, 536-537**
- gettext package (GNU), 536**
- gettype() function, 83**
- getYearEnd() function, 485**
- getYearStart() function, 485**
- giftopnm shell utility, 256**
- global statement**
  - global variables, accessing, 127
  - variable values, remembering between calls, 129-130
- global variables, 77, 127**
- gmdate() function, 184**
- GNU gettext package, 536**
- GRANT command, 31-32**
- greater than (>) operators, 91, 308**
- greater than or equal to (>=) operators, 91, 308**
- group settings (Apache), 56**
- groups file (back-end storage), 495**
- gunzip utility, 17, 40**

## H

- %H format string option (DATE\_FORMAT() function), 337-338**

- %h log-formatting directive, 510-511**
- hard drives, MySQL optimization, 590**
- hardware load balancers, 562**
- Hash Message Authentication Code (HMAC), 576**
- hashes, 503**
- header() function**
  - cookies, setting, 215
  - document formatting, 163
  - redirecting users (forms), 198-200
- headers**
  - cookies, 214
  - From
    - email, 203
    - outgoing email, 200
  - messages (localization), 528-529
  - Reply-to, 203
  - request, 566
  - User-Agent, 498
- heredoc, 378**
- HH:MM:SS time format, 341**
- hidden fields (forms), 197-198**
- HMAC (Hash Message Authentication Code), 576**
- host tables, 29**
- hosting (virtual)**
  - DNS, 564
  - IP-based, 564-565, 570
  - mass, 568-569
  - name-based, 564-570
- HostNameLookups directive, 512, 559**
- hostnames**
  - lookups, 512
  - resolving, 517
- hour functions, 336**
- hour functions, 336**
- .htaccess files, 49, 558**
- htdbm utility, 497**
- htdocs subdirectory, 65**



**HTML (Hypertext Markup Language)**

- calendar form, building, 469-470
- code blocks, 115
- email, formatting, 205
- feedback forms
  - creating, 201-202
  - emailing, 202-203
  - formatting, 205
- file upload forms
  - \$\_FILES superglobal, 206
  - creating, 207
  - scripts, creating, 208-209
  - size, 211
- forms
  - accessing input via arrays, 191-194
  - creating, 189-191
  - input, reading, 190
  - PHP combination, 194
- multiple spaces, displaying, 163
- PHP combination, 71-72
- PHP combination forms
  - calling itself, 194
  - hidden fields, 197-198
  - number-guessing script, 195-196
  - redirecting users, 198-200
  - server headers, 198
- XML, compared, 541

**htpasswd utility, 495**

**htpasswd.exe utility, 495**

**HTTP (Hypertext Transfer Protocol)**

- 1.1, 566
- access, limiting, 501
- headers, 562
- requests, logging, 509
  - files, 514
  - formatting, 510-512
  - host name lookups, 512
  - identity checks, 513
  - images, 514
  - status codes, 513
- secure, 574

- httpd binary, 51

- httpd.conf file, 45

- httpd.exe command, 51

- httpd.pid file, 51

- Hypertext Markup Language.
  - See HTML

- Hypertext Transfer Protocol.
  - See HTTP

**I**

- %i format string option (DATE\_FORMAT() function), 338

- %i log-formatting directive, 511

- I option (httpd/httpd.exe commands), 52

**id fields**

- shopping cart database tables, 452
- store\_categories table, 438

- identd protocol, 513

- identical operators (===), 91

- IdentityCheck directive, 513

- IETF (Internet Engineering Task Force), 574

**if statements**

- code listing, 100
- comparison operators, 90
- else clause, 100-101
- elseif clause, 101-102
- number-guessing script, 196
- redirecting users (forms), 200

- <IfDefine> container, 48

- <IfModule> container, 48

- ImageArc() function, 264

- ImageColorAllocate() function, 263

- ImageCreate() function, 263, 279

- ImageCreateFromGif() function, 271

- ImageCreateFromJpg() function, 271

- ImageCreateFromPng() function, 271

- ImageDestroy() function, 279

- ImageEllipse() function, 264

- ImageFilledArc() function, 266-268

- ImageFilledEllipse() function, 266

- ImageFilledPolygon() function, 266

- ImageFilledRectangle() function, 266

- ImageGif() function, 265

- ImageJpeg() function, 265

- ImageLine() function, 264

- imagemagick() function, 275, 279

- ImagePng() function, 265, 279

- ImagePolygon() function, 264

- ImageRectangle() function, 264

**images****colors**

- allocating, 263
- background, 263
- fills, 266
- RGB values, 262

**creating**

- custom fonts, 279-280
- from existing images, 271-272
- PHP, 261
- scripts, 278-280
- user input, 273-277

**drawing**

- canvas area, 263
- from existing images, 273
- functions, 271
- ImageColorAllocate() function, 263
- ImageCreate() function, 263
- lines, 264-265
- shapes, 264-265
- transparent, 272-273
- x-axis coordinates, 264
- y-axis coordinates, 264

- fonts, 275

- GD graphics library, 262

- JPEG libraries, 263

- logging, 514, 524

- online storefronts, 438, 448

- pie charts, 267
  - 3D, 269-270
  - dynamic data, 281
  - slices, 268
- PNG libraries, 263
- predefined constants, 268
- RGB color values, 262
- stacking, 272-273
- zlib library, 263
- imagestring() function, 273, 277-279**
- imagettftext() function, 275, 279**
- IMAP (Internet Message Access Protocol), 574**
- include\_once statements, 233**
- include\_path directive, 233**
- include statements, 229-230**
  - conditional statements, 232
  - loops, 232
  - performance, 257
  - return values, 231
- included files, 229-230**
  - calling twice, 233
  - conditional statements, 232
  - discussion forums, creating, 418-419
  - loops, 232
  - mailing lists, 375
  - online address books, creating, 390-391
  - performance, 257
  - portability, 233
  - return values, 231
- incorrect permissions (MySQL installation), 26**
- incrementing integers, 89-90**
- INDEX privilege (MySQL), 31**
- index strings, 169**
- infinite loops, 108**
- info option (LogLevel directive), 517**
- inheritance (objects), 155-156**
- INNER JOIN command, 312**
- InnoDB storage engine, 350**
- input (users)**
  - calendars, 467-468
  - elements, escaping, 256
  - forms, 189-194
  - images, creating, 273-277
- input forms**
  - accessing input via arrays, 191-194
  - creating, 189-191
  - discussion forums, creating, 419
  - input, reading, 190
- input scripts, 420-422**
- INSERT command, 303-304**
  - authorized user tables, creating, 502
  - MySQL users, adding, 31
  - syntax, 302
- INSERT privilege (MySQL), 31**
- INSERT statements (MySQL data with PHP), 363**
- INSTALL files (MySQL), 18**
- installation**
  - Apache
    - binary, 39
    - Linux/UNIX, 39-42
    - Mac, 42
    - methods, selecting, 38
    - source code, 38
    - versions, 37-38
    - Windows, 42-44
  - CakePHP, 616
  - CodeIgniter, 617
  - mod\_ssl module
    - UNIX/Linux, 581-582
    - Windows, 580
  - MySQL
    - file downloads, 16
    - Linux/UNIX, 16-18
    - Mac, 18-20
    - troubleshooting, 26-27
    - versions, 15-16
    - Windows, 20-26
  - OpenSSL library
    - UNIX/Linux, 580
    - Windows, 579
  - PHP
    - help, 66-67
    - Linux/UNIX with Apache, 60-62
    - Mac, 63
    - testing, 65-66
    - versions, 59-60
    - Windows, 63-64
  - third-party packages, 5-6
  - XAMPP
    - Linux/UNIX, 6-8
    - Mac OS X, 11-13
    - troubleshooting, 14
    - Windows, 8-11
  - Zend Framework, 615
- instruction terminators, 76**
- INT data type, 298**
- integers, 79, 89-90**
- integrating PHP with Apache**
  - Linux/UNIX, 62-63
  - Windows, 64
- internal caches (MySQL), 595-596**
- internationalization**
  - character sets
    - header messages, 528-529
    - multibyte, 528
    - single-byte, 528
    - unrecognizable characters, 529
  - content translation, compared, 527
  - defined, 527
  - environment modifications
    - Apache, 530
    - MySQL, 531
    - PHP, 530
  - gettext() function, 536-537
  - key aspects, 527
  - locales, 528

- localized page structures, 534
    - language definition file, 531-532
    - language selector, 535
    - locale selection formats, 536
    - string definition file, 533
  - Internet Engineering Task Force (IETF), 574
  - Internet Message Access Protocol (IMAP), 574
  - IP addresses
    - access control, 497
    - reverse DNS lookups, 512
  - IP-based virtual hosting, 564-565, 570
  - irreversible message digests, 576
  - is\_\* functions, 79
  - is\_dir() function, 234
  - is\_executable() function, 235
  - is\_file() function, 234, 238
  - is\_readable() function, 235
  - is\_uploaded\_file() function, 209
  - is\_writable() function, 235
  - isset() function, 468
  - issuers (certificates), 577
  - item\_desc field (store\_items table), 438
  - item\_image field, 438, 448
  - item\_price field (store\_items table), 438
  - item\_title field (store\_items table), 438
  - iterations (loops)
    - defined, 106
    - skipping, 111-112
- J**
- %j format string option (DATE\_FORMAT() function), 337
  - JavaScript Object Notation. See JSON
  - JOIN command, 312
  - Join-Projection normal forms, 294
  - joining tables, 312-314
  - Joomla website, 612
  - JPEG libraries, 263
  - JSON (JavaScript Object Notation), 549
    - API output, 552
    - formatting, 550-552
    - Google search output, 552
    - loading/displaying, 551
    - output, creating, 553
    - Parser, 550
    - website, 550
  - json\_decode() function, 551-553
  - json\_dump.php, 551
  - json\_encode() function, 553
- K**
- %k format string option (DATE\_FORMAT() function), 338
  - KeepAlivetimeout directive, 563
  - key\_buffer\_size parameter, 592
  - key\_read\_requests parameter, 593
  - key\_reads parameter, 593
  - key\_write\_requests parameter, 593
  - key\_writes parameter, 593
  - keys
    - arrays, 140
    - encryption
      - pairs, creating, 582-583
      - public key cryptography, 575-576
      - symmetric cryptography, 574
    - public, 577
  - keywords
    - BINARY, 310
    - public, 151
  - kill command, 52
- L**
- %l format string option (DATE\_FORMAT() function), 338
  - %l log-formatting directive, 510
  - LANGCODE variable, 532
  - language definition file, 531-532
  - language selectors (localization), 535
  - LCASE() function, 329
  - LDAP (Lightweight Directory Access Protocol), 493
  - leading spaces (padding specifiers), 163
  - LEFT() function, 328
  - LEFT JOIN command, 313-314
  - LENGTH() function, 323
  - lengths
    - functions, 322-325
    - names, 345
    - shopping cart database table fields, 452
    - strings, finding, 169
  - less than (<) operators, 91, 308
  - less than or equal to (<=) operators, 91, 308
  - levels (error logging), 516-517
  - lib subdirectory, 65
  - libraries
    - calendar, creating, 483-488
    - component, 615
    - date\_pulldown, 483
    - GD graphics, 262
    - JPEG, 263
    - OpenSSL
      - UNIX/Linux installation, 580
      - website, 579
      - Windows installation, 579
    - PHP Extension Community (PECL), 609
    - PNG, 263
    - SSLeay, 579
    - zlib, 263
  - licenses (Apache), 43
  - Lightweight Directory Access Protocol (LDAP), 493
  - LIKE operator, 309
  - <Limit> container, 501
  - LIMIT clause, 307
  - <LimitExcept> container, 501

- limiting access. See restricting access**
  - lines, drawing, 264-265**
  - links (symbolic), 558**
  - Linux**
    - Apache
      - installation, 39-42
      - modifications, 609
      - starting, 54
      - upgrading, 608
    - apachectl utility, 53
    - httpd binary, 51
    - mod ssl module, installing, 581-582
    - MySQL
      - installation, 16-18
      - upgrading, 607
    - OpenSSL, installing, 580
    - PHP
      - installation with Apache, 60-62
      - Apache integration, 62-63
    - php.ini file, 65
    - server processes, 556
    - XAMPP installation, 6-8
  - list() function, 144**
  - Listen directive, 54, 565**
  - listening addresses, 54**
  - lists (user), accessing, 494**
  - "Little Bobby Tables" comic strip, 362**
  - In command, 558**
  - load distribution, 562**
  - load testing, 559-561**
  - LoadModule directive (SSL configuration), 585**
  - local variables, 77**
  - locales**
    - defined, 528
    - selection formats, 536
  - localization**
    - character sets
      - header messages, 528-529
      - multibyte, 528
      - single-byte, 528
      - unrecognizable characters, 529
    - environment modifications
      - Apache configuration changes, 530
      - MySQL configuration changes, 531
      - PHP configuration changes, 530
    - flags for language selections, 535
    - gettext() function, 536-537
    - internationalization, 527
    - locales
      - defined, 528
      - selection formats, 536
    - numbers/dates/currency, 538
    - page structures
      - language definition file, 531-532
      - language selector, 535
      - locale selection formats, 536
      - string definition file, 533
      - welcome script, 534
  - LOCATE() function, 327**
  - <Location> container, 47**
  - location functions, 327**
  - <Locationmatch> container, 47**
  - locking files, 247-248**
  - LogFormat directive, 512-514**
  - logging**
    - analysis, 518
    - Apache
      - access, 50
      - error, 51
      - httpd.pid file, 51
      - scoreboard file, 51
  - custom**
    - code snippets, creating, 520-521
    - database tables, creating, 519
    - sample reports, 521-523
  - errors**
    - files, 515
    - importance levels, 516-517
    - monitoring, 519
    - programs, 516
    - UNIX syslog daemon, 516
  - files, 514**
  - formatting**
    - CLF (Common Log Format), 512
    - defining, 512
    - directives, 510-511
    - host name lookups, 512
    - identity checks, 513
    - status codes, 513
  - hostname resolution, 517**
  - images, 514, 524**
  - programs, 515**
  - request logs, 509**
  - rotation, 518**
- logical operators, 91-92, 309**
- login forms, 503**
- login scripts, 503-505**
- LogLevel directive, 516-517**
- logresolve utility, 517**
- logresolve.exe utility, 517**
- LONGLOB data type, 301**
- LONGTEXT data type, 301**
- loops, 105**
  - breaking, 109-111
  - do while, 107-108
  - for, 108-109
  - foreach
    - file upload forms, 209
    - multidimensional arrays, 143

- included files, 232
- infinite, 108
- iterations
  - defined, 106
  - skipping, 111-112
- nesting, 112-113
- while, 106-107
  - discussion forum posts, displaying, 429
  - popen() function, 252
- LPAD() function, 326**
- ltrim() function, 174, 325**
- M**
- %M format string option (DATE\_FORMAT() function), 337**
- %m format string option (DATE\_FORMAT() function), 337**
- %m log-formatting directive, 511**
- Mac installations**
  - Apache, 42
  - MAMP package, 5
  - MySQL, 18-20
  - PHP, 63
  - XAMPP, 11-13
- MacPorts website, 63**
- MACs (message authentication codes), 576**
- mail() function, 384**
  - parameters, 203
  - system configuration, 200-201
- mailing lists**
  - bounced messages, 385
  - mailing mechanisms, 381-384
  - MySQL, 27
  - PHP, 67
  - server burden, easing, 385
  - subscription mechanisms
    - include files, creating, 375
    - subscribers table, creating, 374
    - subscription forms, creating, 376-381
- mailing mechanisms, 381-384**
- maintenance**
  - databases, 284
  - releases, 606
- make command**
  - Apache, building, 41
  - PHP installation, 61
- make install command, 41, 61**
- makefiles (Apache installations), 40**
- MAMP installation package, 5**
- managing**
  - logs, 517-519
  - users
    - authentication, 494
    - database file-based authentication, 497
    - file-based authentication, 495
- many-to-many table relationships, 287-288**
- many-to-one mappings (DNS virtual hosting), 564**
- mapping files, 561**
- masks (network), 498**
- mass virtual hosting, 568-569**
- master name tables (online address books), 388-389, 399-401**
- max\_connections variable, 602**
- max\_used\_connections status variable, 602**
- MD5 algorithm, 576**
- meaningOfLife() function, 127**
- MEDIUMBLOB data type, 301**
- MEDIUMINT data type, 298**
- MEDIUMTEXT data type, 301**
- memory**
  - files, mapping, 561
  - MySQL optimization, 590
- menus (online address books), 391**
- message authentication codes (MACs), 576**
- message digests, 576**
- methods**
  - defined, 150
  - GET, 191
  - HTTP, 501
  - objects, 153-154
  - POST, 190-191
- migrating name-based virtual hosts, 570**
- MINUTE() function, 336**
- minute functions, 336**
- mkdir() function, 248-249**
- mktime() function, 184-185**
  - calendar libraries, creating, 485
  - calendar user input, 468
- MMapFile directive, 561**
- mod\_auth\_dbm module, 496-497**
- mod\_auth module, 495-496**
- mod\_authz\_host module**
  - access rules
    - evaluating, 499-500
    - implementing, 497-498
  - clients, 498
  - domain names, 498
  - environment variables, 498
  - IP addresses, 497
- mod\_cache module, 562**
- mod\_deflate module, 563**
- mod\_file\_cache module, 561**
- mod\_ssl module, 579-582**
- mod\_status module, 559**
- Model View Controller. See MVC pattern**
- modifying**
  - Apache
    - installations, 44
    - without upgrading, 608
  - class properties with object methods, 154
  - configuration files, 52
  - data types
    - casting, 82-84
    - settype() function, 81-82
  - environment internationalization configuration changes, 530-531

- images, 271-273
- logs
  - code snippets, creating, 520-521
  - database tables, creating, 519
  - sample reports, 521-523
- object properties, 152
- string text case, 176-177
- strings, 329-330
- tables
  - conditional DELETE command, 321-322
  - DELETE command, 320-321
  - REPLACE command, 319-320
  - UPDATE command, 316-319
- modules**
  - authentication
    - back-end storage, 494
    - denying access, 494
    - directives, 493-494
    - mod\_auth, 495-496
    - mod\_auth\_dbm, 496-497
    - user management, 494
  - mod\_auth\_dbm, 496-497
  - mod\_auth, 495-496
  - mod\_authz\_host
    - clients, 498
    - domain names, 498
    - environment variables, 498
    - evaluating access rules, 499-500
    - implementing access rules, 497-498
    - IP addresses, 497
  - mod cache, 562
  - mod deflate, 563
  - mod file cache, 561
  - mod ssl, 579-582
  - mod status, 559
  - storage, 494
- modulus operators (%), 87**
- Mojibake, 529**
- monitoring error logs, 519**
- MONTH() function, 333**
- month functions, 333**
- month\_select() function, 487**
- MONTHNAME() function, 333**
- \$months variable (calendar libraries), 483**
- move\_uploaded\_file() function, 209**
- MPMs (Multi-Processing Module), 48, 556**
- multibyte character sets, 528**
- multibyte strings website, 530**
- multidimensional arrays**
  - creating, 142-144
  - dimensions, 146
- multiline comments, 72**
- multiple functions, 345**
- multiple spaces (HTML), displaying, 163**
- multiple tables, selecting, 310-312**
- multiplication operators (\*), 87**
- Mutual-Failure argument (Order directive), 500**
- MVC (Model View Controller) pattern, 612-614**
- my.cnf file, 592**
- my-huge.cnf configuration file, 592**
- my-large.cnf configuration file, 592**
- my-medium.cnf configuration file, 592**
- my-small.cnf configuration file, 592**
- MySQL**
  - Announcements list website, 605
  - Configuration wizard, 22
    - character sets, 24
    - completing, 25
    - concurrent connections, 24
    - database usage, 23
    - networking options, 24
    - security, 25
    - server types, 23
  - as service, 25
- connections, 28
- CREATE TABLE command, 301-302
- data
  - inserting with PHP, 363-367
  - retrieving with PHP, 367-369
  - SQL injections, avoiding, 362-363
- data types
  - date/time, 299-300
  - defining, 298
  - numeric, 298-299
  - string, 300-301
- date/time functions
  - arithmetic, 339-341
  - conversion, 342-343
  - current, 341-343
  - days, 331-333
  - formatting, 337-339
  - hours, 336
  - minutes, 336
  - months, 333
  - seconds, 336
  - storing, 187
  - weeks, 334-336
  - years, 334
- DELETE command, 320-321
- field names, 324
- InnoDB storage engine, 350
- INSERT command, 302-304
- installation
  - file downloads, 16
  - Linux/UNIX, 16-18
  - Mac OS X, 18-20
  - troubleshooting, 26-27
  - versions, 15-16
  - Windows, 20-26
- internationalization configuration changes, 531
- mailing list, 27
- Manual website
  - date/time functions, 331

- EXPLAIN command, 595
- FLUSH command, 596
- JOINS, 314
- language-related elements, 531
- MySQL privileges listing, 31
- optimization, 590
- problems and errors, 26
- SHOW command, 597
- SHOW STATUS command, 602
- SHOW VARIABLES command, 602
- startup options, 592
- stored procedures, 355
- subqueries, 315
- transactions, 351
- multiple CPUs, 603
- multiple functions, 345
- obtaining, 16
- optimization, 589
  - benchmarking, 590-591
  - databases/table information, retrieving, 598-599
  - internal caches, 595-596
  - queries, 594-595
  - SHOW command, 596-597
  - startup options, 591-593
  - system status, retrieving, 601-602
  - table structures, 593, 599-601
  - websites, 590
- performance blog, 590
- PHP connections
  - closing, 359
  - creating, 358
  - errors, 359-361
  - queries, executing, 360-361
  - syntax, 358
- PHP functions, 369
- privilege system
  - adding, 31-33

- authentication, 29-31
  - overview, 28-29
  - removing, 33-34
  - tables, 29
- queries, executing with PHP, 360-361
- REPLACE command, 319-320
- running as root, 27, 34
- security, 27-28
- SELECT command, 304-305
  - \* symbol, 305
  - limiting results, 307
  - ordering results, 305-306
  - subqueries, 315
  - syntax, 304
- Setup wizard, 21
- stored procedures, 353-355
- string functions
  - concatenation, 322-325
  - length, 322-325
  - location, 327
  - modification, 329-330
  - padding, 326-327
  - position, 327
  - substring, 328-329
  - trimming, 325
- tables
  - joining, 312-314
  - multiple, selecting, 310-312
- transactions
  - BEGIN command, 352
  - COMMIT command, 350-352
  - defined, 349
  - displaying versus inserting data, 355
  - online storefront example, 351-353
  - ROLLBACK command, 350-352
  - syntax, 350-351
  - website, 351
- UPDATE command, 316-319

- upgrading, 607
- WHERE clauses, 308-309
- Workbench website, 589
- mysqli\_\* functions, 357**
  - mysqli\_close(), 359
  - mysqli\_connect\_error(), 359
  - mysqli\_error(), 361
  - mysqli\_fetch\_arrays(), 368
  - mysqli\_free\_result(), 368
  - mysqli\_insert\_id(), 397, 422
  - mysqli\_num\_rows(), 367-368
  - mysqli\_query(), 361
  - mysqli\_real\_escape\_string(), 397

## N

- \n (newline characters), 113**
- name-based virtual hosting, 564-569**
  - IP-based virtual hosting combination, 570
  - listing, 567
  - migrating, 570
  - request headers, 566
  - ServerAlias directive, 567
  - SSL support, 586
- \$name variable, 483**
- names**
  - characters, 345
  - constants, 94
  - domain, 498
  - error logs, 515
  - functions, 122-123
  - length, 345
  - tables, 301
  - uploaded files, 209
  - variables, 76, 96
- NameVirtualHost directive, 566**
- navigation**
  - breadcrumb trails, 447
  - files, 242
- nesting loops, 112-113**
- Network Information Services (NIS), 493**

**Network News Transfer Protocol (NNTP), 574**

**Network Solutions website, 584**

**networks**

masks, 498

MySQL installation options, 24

settings, 559, 563

**newline characters (\n), 113**

**NIS (Network Information Services), 493**

**nl2br() function, 177**

**NNTP (Network News Transfer Protocol), 574**

**nonequivalence operators (!=), 90**

**normal forms, 289**

additional forms, 294

first, 290

second, 291

third, 291-292

**normalization, 283**

flat tables, 289-290

normal forms, 289

additional forms, 294

first, 290

second, 291

third, 291-292

redundancy, 290

**not equal to operator (!=), 308**

**not operators (!), 92**

**notice option (LogLevel directive), 517**

**now() function, 341-342, 421**

**NULL data types, 79**

**number\_format() function, 538**

**number-guessing script, 195-196**

**numberedHeading() function, 129**

**numbers, localization, 538**

**numeric data types, 298-299**

## O

**%o log-formatting directive, 511**

**object-oriented programming (OOP), 149**

**objects, 79**

constructors, 154

creating, 150-151

declaring, 150

inheritance, 155-156

methods, 153-154

properties, 151-153

declaring, 157

modifying, 152

public keyword, 152

viewing, 152

**OCTET\_LENGTH() function, 323**

**off argument (HostNameLookups directive), 512**

**on argument (HostNameLookups directive), 512**

**ON clause, 313**

**one-to-many mappings (DNS virtual hosting), 564**

**one-to-many table relationships, 286**

**one-to-one mappings (DNS virtual hosting), 564**

**one-to-one table relationships, 285**

**online address books**

birthdays, adding, 414

database tables, creating, 387-390

include files, creating, 390-391

menus, creating, 391

records

adding, 392-398

deleting, 404-406

subentries, adding, 406-413

viewing, 398-404

**online storefronts**

categories of items, displaying, 441-444

database tables

adding records, 439-441

creating, 437-439

field names, 438

store\_categories table, 438

store\_item\_color table, 439

store\_item\_size table, 439

store\_items table, 438

items, displaying, 445-447

shopping carts. See shopping carts

**OOP (object-oriented programming), 149**

**opendir() function, 249**

**opening**

directories, 249

files, 238

appending, 239

failures, 258

reading, 239

writing, 239

pipes, 251

**OpenSSL library, 579**

installing

UNIX/Linux, 580

Windows, 579

website, 579

**operands, 85**

**operating systems**

MySQL optimization, 590

performance limitations, 556-557

**operators**

addition (+), 85

and (&&), 91-92

arithmetic, 86-87

array ([ ]), 140

assignment (=), 77, 86-89

BETWEEN, 309

combined assignment, 88-89

comparison, 90-91

case sensitivity, 310

WHERE clauses, 308

concatenation (.), 87-88

defined, 85

division (/), 87

equal to (=), 308

equivalence operators (==), 90



- greater than operators (>), 91, 308
- greater than or equal to operators (>=), 91, 308
- identical operators (===), 91
- less than operators (<), 91, 308
- less than or equal to operators (<=), 91, 308
- LIKE, 309
- logical, 91-92, 309
- modulus operators (%), 87
- multiplication operators (\*), 87
- not operators (!), 92
- nonequivalence operators (!=), 90
- not equal to operator (!=), 308
- operands, 85
- or operators (||), 91-92
- post-decrement, 89
- post-increment, 89
- precedence, 92-96
- subtraction (-), 87
- ternary (?), 105
- optimization (MySQL), 589**
  - benchmarking, 590-591
  - databases/table information, retrieving, 598-599
  - internal caches, 595-596
  - queries, 594-595
  - SHOW command, 596-597
  - startup options, 591-593
  - system status, retrieving, 601-602
  - table structures, 593, 599-601
  - websites, 590
- OPTIMIZE command, 603**
- OPTIMIZE TABLE SQL command, 593**
- optional arguments, setting, 131**
- options**
  - DATE\_FORMAT() function, 337
  - httpd/httpd.exe binaries, 51
  - LogLevel directive, 516
  - Options directive**
    - mass virtual hosting, 569
    - parameters, 558
  - or operators (||), 91-92**
  - ORDER BY clause**
    - date/time functions, 333
    - DELETE command, 321
    - SELECT command, 305
  - Order directive**
    - access control rules, evaluating, 499
    - Allow,Deny argument, 499
    - Deny,Allow argument, 499
    - Mutual-Failure argument, 500
  - orignum variable (addFive() function), 132**
  - output**
    - binary data, 256
    - JSON, creating, 553
    - processes, reading, 252
  - output() function, 486**
  - overriding directives, 50**
  - ownership (processes), 27**
- P**
  - %p format string option (DATE\_FORMAT() function), 338**
  - packages**
    - third-party installation, 5-6
    - XAMPP
      - download website, 8
      - Linux/UNIX, 6-8
      - Mac OS X, 11-13
      - security, 13-14
      - troubleshooting, 14
      - Windows, 8-11
  - padding**
    - functions, 326-327
    - strings, 162-164
  - padlock icons, 577**
  - parameters**
    - file\_put\_contents() function, 246
    - mail() function, 203
    - MySQL startup, 592
    - Options directive, 558
  - parentheses (), subqueries, 315**
  - parsing XML documents**
    - DOM functions, 544-546
    - SimpleXML functions, 546-549
  - passing data to external applications, 252-253**
  - passthru() function, 256**
  - PASSWORD() function, 502**
  - passwords**
    - authentication, 492
    - encrypting, 495
    - storing, 496, 507
  - PATH environment variables, 64**
  - path fields (cookies), 214**
  - PayPal PayFlo, 463**
  - PCRE (Apache) website, 41**
  - PDO (PHP Data Objects) abstraction layer, 363**
  - PEAR (PHP Extension and Application Repository), 609**
  - PECL (PHP Extension Community Library), 609**
  - per-directory configuration files, 49-50, 57, 558**
  - percent signs (%)**
    - conversion specifications, 160-162
    - log-formatting directives, 510
    - wildcards, 32
  - performance**
    - Apache settings, 558-559
    - databases, 283
    - included files, 257
    - load testing, 559-561
    - operating system limits, 556-557
    - speed, 570
    - tuning
      - abuse prevention, 563-564
      - caching, 562

- load distribution, 562
  - mapping files to memory, 561
  - network settings, 563
  - transmitted data, reducing, 562
- permissions. See privileges**
- personal notes tables (online address books), 390, 401**
- PHP**
- Announcements list website, 605
  - Apache integration, 62-64
  - application frameworks, 614-617
  - changelog website, 606
  - code blocks, 614-616
  - comments, adding, 72-73
  - constants, 94-95
  - cookies, deleting, 217
  - data objects (PDO) abstraction layer, 363
  - data, output, 70
  - data types, 78
    - changing with casting, 82-84
    - changing with settype() function, 81-82
    - standard, 78
    - testing, 79-80, 85
  - dates/times, retrieving, 180
  - directories
    - adding to PATH environment variables, 64
    - creating, 248-249
    - deleting, 249
    - opening, 249
    - reading, 249-251
  - distribution files, downloading, 63
  - expressions, 85-86
  - Extension and Application Repository (PEAR), 609
  - Extension Community Library (PECL), 609
  - extensions, 609
  - file upload forms, 207
  - files
    - appending, 239, 245-246
    - closing, 239
    - locking, 247-248
    - opening, 238
    - reading, 239
    - reading arbitrarily, 241-243
    - reading characters, 243-244
    - reading entire contents, 244-245
    - reading line by line, 240-241
    - writing, 239, 245-247
  - HTML combination, 71-72
  - HTML combination forms
    - calling itself, 194
    - hidden fields, 197-198
    - number-guessing script, 195-196
    - redirecting users, 198-200
    - server headers, 198
  - images. *See* images
  - included files, 229-230
    - calling twice, 233
    - conditional statements, 232
    - loops, 232
    - performance, 257
    - portability, 233
    - return values, 231
  - installation
    - help, 66-67
    - Linux/UNIX with Apache, 60-62
    - Mac, 63
    - testing, 65-66
    - Windows, 63-64
  - integers, incrementing/decrementing, 89-90
  - internationalization configuration changes, 530
  - Linux/UNIX, 62
  - loops. *See* loops
  - mailing lists, 67
- Manual website**
- alternative calendars, 489
  - arrays, 146
  - classes, 150
  - dates/times, 186
  - DOM, 546
  - file locking, 248
  - multibyte strings, 530
  - predefined image-related constants, 268
  - SimpleXML functions, 549
  - strings, 186
- MySQL connections**
- closing, 359
  - creating, 358
  - errors, 359-361
  - queries, executing, 360-361
  - syntax, 358
- MySQL data**
- inserting, 363-367
  - retrieving, 367-369
  - SQL injections, voiding, 362-363
- MySQL functions, 369**
- php.ini file, 61, 65
  - scripts, 67
    - example, 68
    - start/end tags, 68-70
    - text editors, 68, 73
  - statements. *See* statements
- strings**
- argument swapping, 167-168
  - arrays, breaking into, 179
  - case, converting, 176-177
  - cleaning up, 173-174
  - field width specifiers, 164-166
  - indexing, 169
  - lengths, finding, 169
  - nesting functions, 187
  - portions, extracting, 171
  - portions, replacing, 175

- print() function, 160-164
    - storing, 168
    - substrings, finding, 170-171
    - substrings, replacing, 175-176
    - tokenizing, 171-173
    - website resource, 186
    - whitespace
  - text wrapping, removing, 177-179
  - upgrading, 609-610
  - variables. *See* variables
  - versions, 59-60
  - website, 60, 66
  - XML, accessing
    - DOM functions, 544-546
    - SimpleXML functions, 546-549
  - php.ini file, 61, 65, 381**
  - phpinfo() function, 65-66**
  - phpinfo.php file, 65**
  - phpMyAdmin interface, 28**
  - pie charts, creating, 267**
    - 3D, 269-270
    - dynamic data, 281
    - slices, 268
  - pipe symbols (| |), or operators, 91-92**
  - pipes, opening, 251**
  - PNG libraries, 263**
  - pnmscale shell utility, 256**
  - Poedit, 536**
  - Polygons, 264-266**
  - popen() function, 251-253**
  - port values, 54**
  - port variable, 602**
  - portability (included files), 233**
  - ports, binding errors, 55**
  - position functions, 327**
  - post-decrement operators, 89**
  - post-increment operators, 89**
  - POST method, 190-191**
  - \$\_POST superglobal, 77**
  - posts (discussion forums)**
    - adding, 430-433
    - displaying, 426-429
    - first entry, creating, 420-422
  - pound signs (#)**
    - comments, 72
    - directives, 45
    - PHP/Apache integration, 62
  - ppmtogif shell utility, 256**
  - <pre> tags, 163**
  - precedence (operators), 92-96**
  - precision specifiers (string field width), 165**
  - predefined constants, 95**
  - predefined image-related constants, 268**
  - preventing abuse, 563-564**
  - print() function, 70, 120**
  - print\_r() function, 551**
  - printBR() function, 123**
  - printf() function**
    - conversion specifications, 160-162
    - format control string, 160
    - padding specifiers, 162-164
    - type specifiers, 161-162
  - printing cookies, 215**
  - privileges (MySQL)**
    - adding, 31-33
    - authentication, 29-31
    - incorrect, 26
    - overview, 28-29
    - removing, 33-34
    - tables, 29
  - PROCESS privilege (MySQL), 32**
  - processes**
    - output, reading, 252
    - ownership, 27
  - processing, 48-49**
  - procs\_priv table, 29**
  - product price list, formatting, 166**
  - ProgrammableWeb website, 553**
  - Programs, 515-516**
  - prologs (XML documents), 541**
  - properties**
    - classes, 153-154
    - defined, 151
    - objects, 151-153
      - declaring, 157
      - modifying, 152
      - public keyword, 152
      - viewing, 152
  - protocols**
    - HTTP. *See* HTTP
    - identd, 513
    - IMAP, 574
    - LDAP, 493
    - SSL, 574
      - authentication, 576-578
      - certificates. *See* certificates (digital)
      - confidentiality, 574-576
      - configuring, 585
      - connections, 578
      - data integrity, 576
      - mod ssl module, 580-582
      - name-based virtual hosting support, 586
      - OpenSSL, 579-580
      - support module, 579
      - TLS, 574
  - ps command, 27**
  - public keys (certificates), 574-577**
  - public keyword (object properties), 151**
  - putenv() function, 537**
- ## Q
- %q log-formatting directive, 511**
  - queries**
    - MySQL
      - executing with PHP, 360-361
      - optimizing, 594-595

- subqueries, 315
- tables
  - conditions, specifying, 308-309
  - limiting results, 307
  - ordering results, 305-306
  - string comparisons, 309
- question marks (?), ternary operators, 105**
- quotation marks (“”)**
  - MySQL field names, 324
  - strings, escaping, 113
- R**
- %r format string option (DATE\_FORMAT() function), 338**
- %r log-formatting directive, 511**
- RAM disks, 559**
- RC2 algorithm, 574**
- RC4 algorithm, 574**
- readdir() function, 249-251**
- reading**
  - directories, 249-251
  - files, 235, 239
    - arbitrarily, 241-243
    - characters, 243-244
    - entire contents, 244-245
    - line by line, 240-241
    - popen() function, 251
    - process output, 252
- README files (MySQL installation), 18**
- real-time credit card processing (shopping cart database tables), 453**
- realms (authentication), 493**
- records**
  - online address book
    - adding, 392-398
    - birthdays, adding, 414
    - deleting, 404-406
    - subentries, adding, 406-413
    - viewing, 398-404
  - online storefront database
    - tables, adding, 439-441
  - tables
    - adding, 302-304
    - conditional deleting, 321-322
    - conditions, specifying, 308-309
    - deleting, 320-321
    - limiting, 307
    - modifying with REPLACE command, 319-320
    - modifying with UPDATE command, 316-319
    - ordering, 305-306
    - retrieving, 304
    - string comparisons, 309
- rectangles, 264-266**
- redirecting users (forms), 198-200**
- redundancy (normalization), 290**
- registered user sessions, 224-225**
- relationships (tables)**
  - many-to-many, 287-288
  - one-to-many, 286
  - one-to-one, 285
  - types, 284
- RELOAD privilege (MySQL), 32**
- removefromcart.php script, 461-462**
- removing. See deleting**
- REPEAT() function, 330**
- REPLACE command, 319-320**
- REPLACE() function, 330**
- replacing**
  - portions of strings, 175
  - substrings, 175-176
- Reply-to headers (email), 203**
- replytopost.php script, 430-433**
- reports, 521-523**
- request headers, 566**
- request logs, 509**
- \$\_REQUEST superglobal, 78**
- Require directive (authentication), 494**
- require\_once statements, 234**
- require statements, 234**
- reserved constants, 95**
- reset() function, 145**
- resolving hostnames, 517**
- resource data types, 79**
- restricting access**
  - authentication
    - authoritative information, 494
    - back-end storage, 494
    - browsers, 493
    - combining with access control rules, 500
    - database file-based, 496-497
    - defined, 491
    - denying access, 494
    - directives, 493-494
    - etc/passwd file, 507
    - file-based, 495-496
    - realms, 493
    - user lists, 494
    - user management, 494
  - authorization, 492
  - cookies, 502-506
  - HTTP methods, 501
  - rules, 497-500
- resuming sessions, 218-219**
- retrieving MySQL data with PHP, 367-369**
- return statements, 124-125, 168**
- return values (included files), 231**
- reverse DNS lookups, 512**
- REVOKE command, 33**
- RGB color values, 262**
- RIGHT() function, 328**
- RIGHT JOIN command, 314**
- RLimitCPU directive, 557**
- RLimitMem directive, 557**
- RLimitNProc directive, 557**
- rmdir() function, 249**
- robots, 563**
- robots.txt files, 564**

**ROLLBACK command, 350-352**

**root elements (XML documents), 542**

**root users (MySQL), 27, 34**

**rotatelog utility, 515, 518**

**rotatelog.exe utility, 518**

**rotating logs, 518**

**round-robin DNS, 565**

**RPAD() function, 326**

**rtrim() function, 173, 325**

**rules (access control)**

clients, 498

combining with

authentication, 500

domain names, 498

environment variables, 498

evaluating, 499-500

implementing, 497-498

IP addresses, 497

security, 500

**running commands (with functions), 254-256**

## S

**%s format string option (DATE\_FORMAT() function), 338**

**%s log-formatting directive, 511**

**Satisfy directive (access control and authentication combination), 500**

**saving form state, 197-198**

**sayHello() function, 154**

**scalability**

Apache settings, 558-559

load testing, 559-561

operating system limits, 556-557

tuning, 561-564

**ScanErrLog, 519**

**scoreboard files, 51, 559**

**ScoreBoardFile directive, 559**

**script tags, 69**

**ScriptAlias directive (mass virtual hosting), 569**

**scripts**

addentry.php, 392-396, 409-413

addtocart.php, 456-458

configure

Apache installations, 40-41

PHP, 60

control, 53

delentry.php, 404-406

discussion forums

posts, adding, 430-433

posts, displaying, 426-429

topic listing, 423-425

feedback forms, emailing, 202-203

file upload, 208-209

images, creating, 278-280

input, 420-422

localized welcome, 534

PHP, 67-68

comments, adding, 72-73

data, output, 70

example, 68

HTML/PHP combination, 71-72

start/end tags, 68-70

text editors, 68, 73

removefromcart.php, 461-462

replytopost.php, 430-433

selentry.php, 398-404

showcalendar\_withevent.php, 476-478

showcart.php, 458-461

showitem.php

shopping cart storefront integration, 454-456

storefront items, displaying, 445-447

user login, 503-505

**SECOND() function, 336**

**second functions, 336**

**second normal forms, 291**

**sections. See containers**

**Secure Hash (SHA) algorithm, 576**

**security**

abuse prevention, 563-564

access control rules, 500

Apache secure mode, starting, 585

authentication, 492, 573

certificates

chaining, 577

key pairs, creating, 582-583

self-signed, creating, 584

signing, 577

signing requests, 583-584

testing, 577

X.509, 577-578

confidentiality, 573

cookies, 507

data integrity, 573

digest authentication, 492

files, locking, 247-248

hashes, 503

HTTP, 574

MySQL, 25-28

program logging, 515

requirements, 573

software upgrades, 606

**SSL**

authentication, 576-578

certificates. *See* certificates (digital)

confidentiality, 574-576

configuring, 585

connections, 578

data integrity, 576

mod ssl module, 580-582

name-based virtual hosting support, 586

OpenSSL, 579-580

public key cryptography, 575-576

support module, 579

symmetric cryptography, 574

- TLS, 574
- Web, 258
- XAMPP, 13-14
- sel\_\* fields (shopping cart database tables), 452-453**
- sel\_item\_price fields (shopping cart database tables), 453**
- SELECT command**
  - \* symbol, 305
  - limiting results, 307
  - ordering results, 305-306
  - subqueries, 315
  - syntax, 304
  - tables, 310-312
- SELECT privilege (MySQL), 32**
- selentry.php script, 398-404**
- self-signed certificates, creating, 584**
- semicolons (;)**
  - instruction terminators, 76
  - statements, 70
- sending**
  - bulk mail, 383
  - email, 200-201
  - feedback forms via email, 201-205
  - signals, 52
- serialize() function, 221**
- \$\_SERVER superglobal, 78**
- ServerAlias directive, 567**
- ServerName directive**
  - configuration files, checking, 54
  - validity, 57
- ServerRoot directive, 49**
- servers**
  - binary commands, 51-53
  - headers (forms), 198
  - mail burden, easing, 385
  - processes, 556
  - SSL
    - authentication, 576-578
    - certificates. See certificates (digital)
    - confidentiality, 574-576
    - configuring, 585
    - connections, 578
    - data integrity, 576
    - mod ssl module, 580-582
    - name-based virtual hosting support, 586
    - OpenSSL, 579-580
    - support module, 579
  - TLS, 574
  - types, 23
  - virtual, 47
- services**
  - MySQL as, 25
  - Network Information (NIS), 493
- session\_destroy() function, 223-224**
- session\_id fields (shopping cart database tables), 452**
- session\_id() function, 218**
- session\_save\_path() function, 220**
- session\_set\_save\_handler() function, 218**
- session\_start() function, 218, 532**
- \$\_SESSION superglobal, 78, 219**
- sessions**
  - destroying, 223-224
  - ids, accessing, 218
  - overview, 217
  - pitfalls, 226
  - registered users, 224-225
  - resuming, 218-219
  - session\_set\_save\_handler() function, 218
  - starting, 218-219
  - state, 218
  - user preferences, 225
  - variables
    - accessing, 219-223
    - adding to arrays, 221
    - removing, 224
    - storing, 219
- Set-Cookie header, 214**
- SET data type, 301**
- set\_time\_limit() function, 383**
- setcookie() function, 215**
- setDate\_array() function, 484**
- setDate\_global() function, 485**
- setDate\_timestamp() function, 484**
- setlocale() function, 537**
- setName() function, 154**
- settype() function, 81-82**
- setYearEnd() function, 485**
- setYearStart() function, 485**
- SHA (Secure Hash) algorithm, 576**
- shading pie charts, 269**
- shapes, drawing, 264-265**
- shell utilities, 256**
- shipping addresses (shopping cart database tables), 453**
- shopping carts**
  - checkout actions, 463-464
  - checkout forms, creating, 463
  - database tables, 451-453
  - items
    - adding, 456-458
    - inventory, 465
    - removing, 461-462
  - storefront integration, 453-456
  - viewing, 458-461
- short tags, 69**
- SHOW COLUMNS command, 599**
- SHOW command, 596-597**
- SHOW CREATE TABLE command, 599**
- SHOW DATABASES command, 598-599**
- SHOW GRANTS command, 597**
- SHOW INDEX command, 600**
- SHOW STATUS command, 593, 601-602**
- SHOW TABLE STATUS command, 600**
- SHOW VARIABLES command, 601-602**

- showcalendar\_withevent.php script, 476-478
- showcart.php script, 458-461
- showitem.php script
  - shopping cart storefront integration, 454-456
  - storefront items, displaying, 445-447
- shuffle() function, 145
- SHUTDOWN privilege (MySQL), 32
- signals, sending, 52
- signatures (certificates), 577
- signing requests (certificates), 583-584
- SimpleXML functions, 546-549
- simplexmlexample.php, 549
- single-byte character sets, 528
- single-line comments, 72
- size
  - cookies, 213
  - files
    - determining, 235-236
    - upload forms, 211
- sizeof() function, 144
- slow\_queries status variable, 602
- SMALLINT data type, 298
- software
  - Apache installation, configuring, 40-41
  - load balancers, 562
  - upgrades
    - Apache, 608-609
    - maintenance releases, 606
    - MySQL, 607
    - PHP 609-610
    - security fixes, 606
    - version changes, 606
    - websites, 605
    - when to upgrade, 606-607
- Solaris, 556-557
- source code (Apache), 38-40
- source files (directories), 60
- spaces (text), 163
- special characters, 163
- speed (performance), 570
- sprintf() function, 168, 487
- SQL injections, avoiding, 362-363
- SSL (Secure Sockets Layer), 574
  - authentication, 576-578
  - certificates
    - chaining, 577
    - key pairs, creating, 582-583
    - self-signed, creating, 584
    - signing, 577
    - signing requests, 583-584
    - testing, 577
    - X.509, 577-578
  - confidentiality, 574-576
  - configuring, 585
  - connections, 578
  - data integrity, 576
  - mod ssl module, 580-582
  - name-based virtual hosts, 566, 586
  - OpenSSL, 579-580
  - support module, 579
- SSLCertificateFile directive, 585
- SSLCertificateKeyfile directive, 585
- SSLey library, 579
- SSLEngine directive, 585
- stacking images, 272-273
- standard data types, 78
- standard tags, 69
- start tags, 68-70, 73
- \$start variable (calendars), 473
- starting
  - Apache
    - access denied, 56
    - binding to ports
      - permissions, 55
    - browsers, 54
    - configuration files, checking, 53-54
    - existing web servers, 55
    - group settings, 56
    - Linux/UNIX, 54
    - secure mode, 585
    - Windows, 54
  - comments, 72
  - MySQL, 27-28
  - sessions, 218-219
  - statements, 68-70
- startup options (MySQL), 591-593
- state
  - forms, saving, 197-198
  - sessions, storing, 218
- statements. *See also* commands
  - break, 109-111
  - conditional, 232
  - continue, 111-112
  - do while, 107-108
  - echo, 114
    - multidimensional arrays, 144
    - PHP scripts, 70
  - ending, 70
  - exit, 200
  - for, 108-111
  - foreach, 209
  - function, 121
  - global
    - global variables,
      - accessing, 127
    - variable values, remembering
      - between calls, 129-130
  - if
    - code listing, 100
    - comparison operators, 90
    - else clause, 100-101
    - elseif clause, 101-102
    - number-guessing script, 196
    - redirecting users (forms), 200
  - include, 229-230
    - conditional statements, 232
    - loops, 232
    - performance, 257
    - return values, 231

- include\_once, 233
- INSERT, 363
- PHP, 76
- require, 234
- require\_once, 234
- return, 124-125, 168
- starting/ending, 68-70
- static, 130
- switch, 103-104
- while, 106-107, 241
- static statement, 130**
- status**
  - codes, logging, 513
  - files, checking, 235
  - settings, 559
  - system, retrieving, 601-602
- storage**
  - back-end, 494-496
  - data (XML), 553
  - formatted strings, 168
  - logs, 514
  - passwords, 496, 507
  - sessions, 218-219
- stored procedures, 353-355**
- storefronts**
  - categories of items, displaying, 441-444
  - database tables
    - adding records, 439-441
    - creating, 437-439
    - field names, 438
    - store\_categories table, 438
    - store\_item\_color table, 439
    - store\_item\_size table, 439
    - store\_items table, 438
  - items, displaying, 445-447
  - shopping carts. See shopping carts
- store\_categories table, 438-439**
- store\_item\_color table, 438-441**
- store\_item\_size table, 438-440**
- store\_items table, 438-440**
- str\_replace() function, 175-176**
- string data types, 300-301**
- \$string variable (strip\_tags() function), 174**
- strings, 79**
  - arrays, breaking into, 179
  - cleaning up, 173-174
  - definition file, 533
  - \$display\_block, 399
  - escaping quotation marks, 113
  - formatting, 160
    - argument swapping, 167-168
    - field width specifiers, 164-166
    - printf() function, 160-164
    - storing, 168
  - functions
    - concatenation, 322-325
    - length, 322-325
    - location, 327
    - modification, 329-330
    - padding, 326-327
    - position, 327
    - substring, 328-329
    - trimming, 325
  - indexing, 169
  - lengths, finding, 169
  - log formats, 510-511
  - nesting functions, 187
  - portions
    - extracting, 171
    - replacing, 175
  - substrings
    - finding, 170-171
    - replacing, 175-176
  - tags, removing, 174
  - text
    - case, converting, 176-177
    - wrapping, 177-179
  - tokenizing, 171-173
  - website resources, 186
  - whitespace, removing, 173
- strip\_tags() function, 174**
- stripslashes() function, 425**
- strlen() function, 169**
- strpos() function, 170-171**
- strstr() function, 170**
- strtok() function, 171-173, 179**
- strtolower() function, 176**
- strtoupper() function, 120**
- strtoupper\_replace() function, 176**
- subdirectories, 65, 592**
- subentries (online address book records), adding, 406-413**
- subexpressions, 86**
- subjects (certificates), 577**
- subqueries, 315**
- subscribers tables, creating, 374**
- subscription forms, creating, 376-381**
- subscription mechanisms**
  - include files, creating, 375
  - subscribers table, creating, 374
  - subscription forms, creating, 376-381
- substr() function, 171**
- substr\_replace() function, 175**
- SUBSTRING() function, 328**
- substring functions, 328-329**
- substrings**
  - finding, 170-171
  - replacing, 175-176
- subtraction operators (-), 87**
- superglobal variables**
  - \$\_COOKIE, 77, 215
  - \$\_ENV, 78
  - \$\_FILES, 78, 206
  - \$\_GET, 77
  - \$\_POST, 77
  - \$\_REQUEST, 78
  - \$\_SERVER, 78
  - \$\_SESSION, 78, 219
- support-files subdirectory (MySQL), 592**



swapping arguments, 167-168

switch statements, 103-104

switching flow

if statements

code listing, 100

else clause, 100-101

elseif clause, 101-102

switch statements, 103-104

ternary operators, 105

symbolic links, 558

Symfony framework website, 614

SymLinksIfOwnerMatch parameter  
(Options directive), 558

symmetric cryptography, 574

SYSDATE() function, 342

syslog daemon (error logging),  
515-516

system status, retrieving, 601-602

system() function, 255

## T

%T format string option (DATE\_  
FORMAT() function), 338

%t log-formatting directive, 510-511

table-cache parameter, 593

table\_type variable, 602

tables

authorized users, creating,  
502-503

calendar, creating, 471-474

calendar\_events, 474

creating, 301-302

custom logs

code snippet, 520-521

creating, 519

sample reports, 521-523

discussion forum database,  
creating, 417-418

flat, 289-290

information, retrieving, 598-599

joining, 312-314

modifying

REPLACE command, 319-320

UPDATE command, 316-319

multiple, selecting, 310-312

MySQL privileges, 29

names, 301

online address books, creating,  
387-390

online storefront databases

adding records, 439-441

creating, 437-439

field names, 438

store\_categories table, 438

store\_item\_size table, 439

store\_items table, 438

queries

conditions, specifying,  
308-309

limiting, 307

ordering, 305-306

string comparisons, 309

records

adding, 302-304

conditional deleting, 321-322

deleting, 320-321

retrieving, 304

relationships

many-to-many, 287-288

one-to-many, 286

one-to-one, 285

types, 284

shopping cart databases

adding items to cart,  
456-458

cart, viewing, 458-461

checkout actions, 463-464

checkout forms, creating, 463

date items were added to  
cart field, 452

fields, 451-453

item inventory, 465

real-time credit card  
processing, 453

removing items from cart,  
461-462

selections, holding, 452

shipping addresses, 453

storefront integration,  
453-456

users, identifying, 452

structure

information, retrieving,  
599-601

optimizing, 593

subqueries, 315

subscribers, creating, 374

tables\_priv table, 29

tags

ASP, 69

end, 68-70, 73

<pre>, 163

script, 69

short, 69

standard, 69

start, 68-70, 73

strings, removing, 174

XML, 543

tagWrap() function, 134

tail command-line utility, 519

tar utility, 17

tarball, 40

targets (makefiles), 40

telephone tables (online address  
books), 389

ternary operators (?), 105

test() function, 126

testing

auth cookies, 506

certificates, 577

data types, 79-80, 85

dates, 185

files, 236-238

dates/times, 236

executability, 235

existence, 234

file/directory confirmation,  
234

- readability, 235
- size, 235-236
- status, 235
- writability, 235
- function availability, 133-134
- load, 559-561
- PHP installations, 65-66
- text**
  - documents, formatting, 163
  - images, creating, 279-280
  - strings
    - converting, 176-177
    - wrapping, 177-179
- TEXT data type, 300**
- text editors (PHP scripts), 68, 73**
- textdomain() function, 537**
- Thawte, 584**
- third normal forms, 291-292**
- third-party installation packages, 5-6**
- TIME data type, 300**
- time() function, 180, 216**
- TimeOut directive, 563**
- times/dates**
  - calendars
    - events, adding, 474-482
    - HTML forms, building, 469-470
    - libraries, creating, 483-488
    - tables, creating, 471-474
    - user input, 467-468
  - current, retrieving, 180
  - data types, 299-300
  - databases, 187
  - dates, testing, 185
  - files, 236
  - functions
    - arithmetic, 339-341
    - conversion, 342-343
    - current, 341-343
    - days, 331-333
    - formatting, 337-339
    - hours, 336
    - minutes, 336
    - months, 333
    - seconds, 336
    - weeks, 334-336
    - years, 334
  - HH:MM:SS time format, 341
  - timestamps
    - converting to dates with date() function, 182-184
    - converting to dates with getdate() function, 180-182
    - creating, 184-185
    - defined, 180
    - UNIX epoch, 180
    - website resources, 186
    - YYYY-MM-DD format, 341
- TIMESTAMP data type, 300**
- \$timestamp variable (calendar libraries), 483**
- timestamps**
  - converting to dates
    - date() function, 182-184
    - getdate() function, 180-182
  - creating, 184-185
  - defined, 180
- TINYBLOB data type, 300**
- TINYINT data type, 298**
- TINYTEXT data type, 300**
- TLS (Transport Layer Security), 574**
- tokenizing strings, 171-173**
- topics (discussion forums)**
  - first entry, creating, 420-422
  - forms, 419
  - lists, displaying, 423-426
  - posts
    - adding, 430-433
    - displaying, 426-429
  - scripts, 420-422
- touch() function, 238**
- tracking client requests, 50**
- TRAILING function, 326**
- transactions**
  - BEGIN command, 352
  - COMMIT command, 350-352
  - defined, 349
  - displaying versus inserting data, 355
  - online storefront example, 351-353
  - ROLLBACK command, 350-352
  - syntax, 350-351
  - website, 351
- TransferLog directive, 514**
- Transifex, 536**
- translation catalog files, 536**
- transmitted data, reducing, 562**
- transparent images, 272-273**
- Transport Layer Security (TLS), 574**
- trim() function, 173**
- trimming functions, 325**
- Triple DES algorithm, 574**
- troubleshooting**
  - Apache startup, 55-56
  - installations, 14
  - MySQL installations, 26-27
- tuning performance**
  - abuse prevention, 563-564
  - caching, 562
  - load distribution, 562
  - mapping files to memory, 561
  - network settings, 563
  - transmitted data, reducing, 562
- TYPE argument (file upload forms), 207**
- type specifiers (printf() function), 161-162**
- U**
  - %U format string option (DATE\_FORMAT() function), 337
  - %u log-formatting directive, 510-511
  - UCASE() function, 329
  - ucfirst() function, 177
  - ucwords() function, 176

**ulimit command, 556**

**uncompressing Apache source code, 40**

**underline() function, 134**

**“Understanding Model-View-Controller” blog, 614**

## UNIX

### Apache

- installation, 39-42
- modifications, 609
- starting, 54
- upgrading, 608

apachectl utility, 53

column command, 252-253

directories, listing, 254

epoch, 180

FROM\_UNIXTIME() function, 342

httpd binary, 51

In command, 558

logresolve utility, 517

mod ssl module, installing, 581-582

### MySQL

- installation, 16-18
- upgrading, 607

OpenSSL, installing, 580

PHP with Apache, 60-63

php.ini file, 65

rotatelogs utility, 518

syslog daemon, 515-516

tail command-line utility, 519

ulimit command, 556

UNIX\_TIMESTAMP() function, 342

who command output, reading, 252

XAMPP installation, 6-8

**UNIX\_TIMESTAMP() function, 342**

**unlink() function, 238**

**unrecognizable characters, 529**

**unsubscribe forms, creating, 378-381**

## UPDATE command

conditional, 317

existing values, 318-319

subqueries, 315

tables, 316-317

**UPDATE privilege (MySQL), 32**

**updating tables, 316-319**

## upgrades

Apache, 608-609

MySQL, 607

PHP, 609-610

Software, 605-607

**uptime status variable, 602**

## URLs

directives, applying, 47

form values, viewing, 210

**User-Agent headers, environment variable access control, 498**

## user-defined functions

calling, 121

values, returning, 124-125

## users

authorization tables, creating, 502-503

databases, adding/deleting, 497

login forms, 503

login scripts, 503-505

### input

calendars, 467-468

elements, escaping, 256

forms, 189-194

images, creating, 273-277

lists, authentication, 494

### managing

authentication, 494

database file-based authentication, 497

file-based authentication, 495

MySQL, adding, 31-33

names, authentication, 492

redirecting forms, 198-200

root, 27, 34

### sessions

destroying, 223-224

ids, accessing, 218

overview, 217

pitfalls, 226

registered users, 224-225

resuming, 218-219

session\_set\_save\_handler() function, 218

starting, 218-219

state, 218

user preferences, 225

variables, 219-224

tables, 29

**users file (back-end storage), 495**

**usr/local/apache2 directory, 41**

**usr/local/php/lib directory, 65**

**usr/local/src directory, 60**

**usr/src directory, 60**

## utilities

apachectl, 53

giftopnm shell, 256

gunzip, 17, 40

htdbm, 497

htpasswd/htpasswd.exe, 495

logresolve, 517

logresolve.exe, 517

pnmscale shell, 256

ppmtogif shell, 256

rotatelogs, 515, 518

rotatelogs.exe, 518

tail command-line, 519

tar, 17

## V

**%V format string option (DATE\_FORMAT() function), 337**

**%V log-formatting directive, 511**

**v option (httpd/httpd.exe commands), 52**

**validating files, 234-236**

**value directives, 65**

## values

arguments, 130-132

- arrays, 140
- functions, returning, 124-125
- port, 54
- return, 231
- variables, 76, 129-130

**VARCHAR(M) data type, 300**

**variables**

- assignment operator (=), 77
- availability, 77
- casting, 82-84
- CHARSET, 532
- \$check\_res, 379
- \$COOKIE, 215
- \$count, 473
- declaring, 76
- defined, 75
- environment
  - access control, 498
  - HTTP\_COOKIE, 215
  - PATH, 64
- \$file\_array, 209
- \$file\_dir, 209
- \$file\_name, 209
- functions
  - declaring outside, 126
  - declaring within, 125-126
  - global access, 126-128
  - references, passing, 132-133
  - values between calls, remembering, 129-130
- global, 77, 127
- integer, 89-90
- LANGCODE, 532
- local, 77
- \$month, 483
- \$name, 483
- names, 76, 96
- orignum, 132
- sessions, 219
  - accessing, 219-223
  - adding to arrays, 221
  - removing, 224
  - storing, 219
- SHOW VARIABLES command, 602
- \$start, 473
- status, 602
- \$string, 174
- superglobal, 77
  - \$\_COOKIE, 77, 215
  - \$\_ENV, 78
  - \$\_FILES, 78, 206
  - \$\_GET, 77
  - \$\_POST, 77
  - \$\_REQUEST, 78
  - \$\_SERVER, 78
  - \$\_SESSION, 78, 219
  - \$timestamp, 483
  - values, 76
- VeriSign, 584**
- version variable, 602**
- versions**
  - Apache, 37-38
  - MySQL, 15-16
  - PHP, 59-60
- viewing. See displaying**
- virtual hosting, 564**
  - DNS, 564
  - IP-based, 564-565, 570
  - mass, 568-569
  - name-based, 564-570
    - listing, 567
    - migrating, 570
    - request headers, 566
    - ServerAlias directive, 567
    - SSL support, 586
- virtual servers, specifying, 47**
- VirtualDocumentRoot directive, 568**
- VirtualDocumentRootIP directive, 569**
- <VirtualHost> containers, 47, 565**
- VirtualScriptAlias directive, 569**
- VirtualScriptAliasIP directive, 569**

## W

- %W format string option (DATE\_FORMAT() function), 337**
- WAMP installation package, 5**
- warn option (LogLevel directive), 517**
- web**
  - crawlers, 563
  - pages (localization)
    - flags for language selections, 535
    - language definition file, 531-532
    - language selector, 535
    - local selection formats, 536
    - string definition file, 533
    - welcome script, 534
  - security, 258
  - server activity, logging
    - analysis, 518
    - code snippets, creating, 520-521
    - database tables, creating, 519
    - errors, 515-519
    - files, 514
    - formatting, 510-513
    - hostname lookups, 512
    - hostname resolution, 517
    - identity checks, 513
    - images, 514, 524
    - programs, 515
    - request logs, creating, 509
    - rotation, 518
    - sample reports, 521-523
    - status codes, 513
  - spiders, 563
- Webalizer, 519**
- websites**
  - accept mechanisms, 559
  - alternative calendars, 489
  - Apache, 37-39
  - APR, 41

- based content
  - negotiation, 530
- Caching Guide, 562
- News and Announcements
  - list, 605
- application frameworks listing
  - (Wikipedia), 614
- arrays, 144-146
- Authorize.Net, 463
- awstats, 519
- bindtextdomain() function, 537
- CakePHP, 616
- CAPTCHAs, 279
- CodeIgniter, 616-617
- Common Log Format documentation, 512
- dates/times resources, 186
- Debian file downloads, 17
- directives, 46
- DOM, 544
- Drupal, 612
- echo() function, 120
- file descriptors, 557
- file locking, 248
- Get Localization, 536
- GNU gettext package, 536
- heredoc, 378
- identd protocol, 513
- InnoDB storage engine, 350
- Joomla, 612
- JPEG libraries, 263
- JSON, 550
- MacPorts, 63
- MAMP installation package, 5
- Mojibake, 529
- MVC pattern, 614
- MySQL
  - Announcements, 605
  - file downloads, 16
  - mailing list, 27
  - optimization, 590
- Performance blog, 590
- upgrades, 607
- Workbench, 589
- MySQL Manual
  - date/time functions, 331
  - EXPLAIN command, 595
  - FLUSH command, 596
  - JOINS, 314
  - language-related
    - elements, 531
  - optimization, 590
  - privileges listing, 31
  - problems and errors, 26
  - SHOW command, 597
  - SHOW STATUS command, 602
  - SHOW VARIABLES
    - command, 602
  - startup options, 592
  - stored procedures, 355
  - subqueries, 315
  - transactions, 351
- mysqli\_\* functions, 357, 369
- Network Solutions, 584
- normal forms, 294
- OpenSSL library, 579
- PayPal PayFlow, 463
- PCRE (Apache), 41
- PDO abstraction layer, 363
- PHP, 60, 66
  - Announcements, 605
  - changelog, 606
  - directory, adding to PATH
    - environment variables, 64
  - mailing lists, 67
  - text editors, 68
- PHP Manual
  - alternative calendars, 489
  - arrays, 146
  - classes, 150
  - dates/times, 186
  - DOM, 546
  - file locking, 248
  - multibyte strings, 530
  - predefined image-related
    - constants, 268
  - SimpleXML functions, 549
  - strings, 186
- phpMyAdmin interface, 28
- PNG libraries, 263
- Poedit, 536
- predefined constants, 95
- print() function, 120
- ProgrammableWeb, 553
- reserved constants, 95
- RGB color values, 262
- robots.txt, 564
- ScanErrLog, 519
- setlocale() function, 537
- SimpleXML functions, 549
- software upgrades, 605
- SQL injections, 362-363
- SSL with virtual hosts, 566
- string resource, 186
- Symfony framework, 614
- textdomain() function, 537
- Thawte, 584
- Transifex, 536
- VeriSign, 584
- WAMP installation package, 5
- web security, 258
- Webalizer, 519
- Win32 distribution notes, 43
- WordPress, 612
- Wusage, 519
- XAMPP download, 8
- XML document specification, 542
- Yii framework, 614
- Zend Framework, 615
- zlib library, 263
- week functions, 334-336**
- WEEKDAY() function, 331**
- WHERE clause, 308-309**

**while loops, 106-107, 241**

- discussion forum posts,
- displaying, 429
- popen() function, 252

**whitespace (strings), 173****who command output, reading, 252****Wikipedia application framework listing, 614****wildcards (\*/%), 32****Win32 distribution notes, 43****Windows**

- Apache
    - installation, 42-44
    - modifications, 609
    - starting, 54
    - upgrading, 608
  - htpasswd utility, 495
  - httpd.exe binary, 51
  - logresolve.exe utility, 517
  - mod ssl module, installing, 580
  - MySQL
    - installation, 20-26
    - upgrading, 607
  - OpenSSL, installing, 579
  - PHP
    - Apache integration, 64
    - installation, 63-64
    - upgrading, 609
  - rotatelogs.exe utility, 518
  - WAMP installation package, 5
  - XAMPP, 8-11
- wizards (MySQL)**
- Configuration, 22-25
  - Setup, 21
- WordPress website, 612**
- wordwrap() function, 178-179**
- Workbench (MySQL), 589**
- “The World Wide Web Security FAQ” website, 258**

**wrapping string text, 177-179****writing files, 235, 239**

- file\_put\_contents() function, 246-247
- fopen() function, 245
- fwrite() function, 246

**Wusage, 519****X****x-axis coordinates, 264****%x format string option (DATE\_FORMAT() function), 337****%X log-formatting directive, 511****X.509 certificates, 577****XAMPP**

- download website, 8
- Linux/UNIX, 6-8
- Mac OS X, 11-13
- security, 13-14
- troubleshooting, 14
- Windows, 8-11

**XML (Extensible Markup Language), 541**

- capabilities, 543
- data storage, 553
- defined, 541
- documents, 541-543
  - case sensitivity, 543
  - children, 542
  - content structure, 542
  - parsing with DOM functions, 544-546
  - parsing with SimpleXML functions, 546-549
  - prologs, 541
  - root elements, 542
  - sample, 542
  - tags, 543
  - XML specification, 542

**HTML, compared, 541****PHP access**

- DOM functions, 544-546
- SimpleXML functions, 546-549

**xor operators, 92****Y****y-axis coordinates, 264****%y format string option (DATE\_FORMAT() function), 337****%y log-formatting directive, 511****YEAR data type, 300****YEAR() function, 334****year functions, 334****year\_select() function, 487****Yii framework website, 614****YYYY-MM-DD date format, 341****Z****Zend engine, 615****Zend Framework, 615****zlib library, 263**