LEARNING MADE EASY

# PHP, MySQL® & JavaScript® ALL-IN-ONE

for dummies®
A Wiley Brand

7 Books in one!

Richard Blum

# PHP, MySQL® & JavaScript®

ALL-IN-ONE

by Richard Blum

for dummies®
A Wiley Brand

**PHP, MySQL® & JavaScript® All-in-One For Dummies®**

Published by: **John Wiley & Sons, Inc.,** 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com

Copyright © 2018 by John Wiley & Sons, Inc., Hoboken, New Jersey

Published simultaneously in Canada

Library of Congress Control Number: 2018933793

ISBN 978-1-119-46838-7 (pbk); ISBN 978-1-119-46833-2 (ebk); ISBN 978-1-119-46837-0 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

# Contents at a Glance

**Introduction** . . . . . . . . 1

**Book 1: Getting Started with Web Programming** . . . . . . . . 5

CHAPTER 1: Examining the Pieces of Web Programming . . . . . . . . 7
CHAPTER 2: Using a Web Server . . . . . . . . 27
CHAPTER 3: Building a Development Environment . . . . . . . . 51

**Book 2: HTML5 and CSS3** . . . . . . . . 71

CHAPTER 1: The Basics of HTML5 . . . . . . . . 73
CHAPTER 2: The Basics of CSS3 . . . . . . . . 103
CHAPTER 3: HTML5 Forms . . . . . . . . 135
CHAPTER 4: Advanced CSS3 . . . . . . . . 157
CHAPTER 5: HTML5 and Multimedia . . . . . . . . 177

**Book 3: JavaScript** . . . . . . . . 195

CHAPTER 1: Introducing JavaScript . . . . . . . . 197
CHAPTER 2: Advanced JavaScript Coding . . . . . . . . 223
CHAPTER 3: Using jQuery . . . . . . . . 243
CHAPTER 4: Reacting to Events with JavaScript and jQuery . . . . . . . . 263
CHAPTER 5: Troubleshooting JavaScript Programs . . . . . . . . 283

**Book 4: PHP** . . . . . . . . 301

CHAPTER 1: Understanding PHP Basics . . . . . . . . 303
CHAPTER 2: PHP Flow Control . . . . . . . . 325
CHAPTER 3: PHP Libraries . . . . . . . . 349
CHAPTER 4: Considering PHP Security . . . . . . . . 375
CHAPTER 5: Object-Oriented PHP Programming . . . . . . . . 395
CHAPTER 6: Sessions and Carts . . . . . . . . 419

**Book 5: MySQL** . . . . . . . . 443

CHAPTER 1: Introducing MySQL . . . . . . . . 445
CHAPTER 2: Administering MySQL . . . . . . . . 465
CHAPTER 3: Designing and Building a Database . . . . . . . . 489
CHAPTER 4: Using the Database . . . . . . . . 513
CHAPTER 5: Communicating with the Database from PHP Scripts . . . . . . . . 541

**Book 6: Creating Object-Oriented Programs** . . . . . . . . . . . . . . . 561
CHAPTER 1: Designing an Object-Oriented Application. . . . . . . . . . . . . . . . . . . . . . . . 563
CHAPTER 2: Implementing an Object-Oriented Application . . . . . . . . . . . . . . . . . . . . 593
CHAPTER 3: Using AJAX . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 619
CHAPTER 4: Extending WordPress . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 651

**Book 7: Using PHP Frameworks** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 681
CHAPTER 1: The MVC Method . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 683
CHAPTER 2: Selecting a Framework . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 695
CHAPTER 3: Creating an Application Using Frameworks. . . . . . . . . . . . . . . . . . . . . . . 715

**Index** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 735

# Table of Contents

INTRODUCTION . . . . . 1

About This Book. . . . . 1

Foolish Assumptions. . . . . 2

Icons Used in This Book . . . . . 3

Beyond the Book . . . . . 3

Where to Go from Here . . . . . 3

BOOK 1: GETTING STARTED WITH WEB PROGRAMMING. . . . . 5

CHAPTER 1: Examining the Pieces of Web Programming . . . . . 7

Creating a Simple Web Page . . . . . 7

Kicking things off with the World Wide Web . . . . . 8

Making sense of markup languages . . . . . 9

Retrieving HTML documents . . . . . 10

Styling . . . . . 14

Creating a Dynamic Web Page . . . . . 17

Client-side programming . . . . . 19

Server-side programming . . . . . 21

Combining client-side and server-side programming . . . . . 24

Storing Content . . . . . 25

CHAPTER 2: Using a Web Server . . . . . 27

Recognizing What’s Required. . . . . 27

The web server. . . . . 28

The PHP server. . . . . 29

The database server . . . . . 30

Considering Your Server Options . . . . . 31

Using a web-hosting company . . . . . 32

Building your own server environment . . . . . 33

Using premade servers. . . . . 37

Tweaking the Servers . . . . . 41

Customizing the Apache Server . . . . . 41

Customizing the MySQL server . . . . . 44

Customizing the PHP server. . . . . 46

CHAPTER 3: Building a Development Environment. . . . . 51

Knowing Which Tools to Avoid . . . . . 51

Graphical desktop tools . . . . . 52

Web-hosting sites . . . . . 52

Word processors . . . . . 53

Working with the Right Tools . . . . . 53
Text editors . . . . . 53
Program editors . . . . . 61
Integrated development environments . . . . . 64
Browser debuggers . . . . . 67

**BOOK 2: HTML5 AND CSS3 . . . . . 71**

CHAPTER 1: **The Basics of HTML5 . . . . . 73**
Diving into Document Structure . . . . . 73
Elements, tags, and attributes . . . . . 73
Document type . . . . . 75
Page definition . . . . . 76
Page sections . . . . . 78
Looking at the Basic HTML5 Elements . . . . . 81
Headings . . . . . 81
Text groupings . . . . . 82
Breaks . . . . . 84
Marking Your Text . . . . . 85
Formatting text . . . . . 85
Using hypertext . . . . . 86
Working with Characters . . . . . 90
Character sets . . . . . 90
Special characters . . . . . 91
Making a List (And Checking It Twice) . . . . . 92
Unordered lists . . . . . 92
Ordered lists . . . . . 93
Description lists . . . . . 95
Building Tables . . . . . 96
Defining a table . . . . . 96
Defining the table's rows and columns . . . . . 97
Defining the table headings . . . . . 99

CHAPTER 2: **The Basics of CSS3 . . . . . 103**
Understanding Styles . . . . . 103
Defining the rules of CSS3 . . . . . 104
Applying style rules . . . . . 110
Cascading style rules . . . . . 111
Styling Text . . . . . 112
Setting the font . . . . . 112
Playing with color . . . . . 116
Working with the Box Model . . . . . 119
Styling Tables . . . . . 121
Table borders . . . . . 122
Table data . . . . . 123

Positioning Elements .....125
Putting elements in a specific place .....128
Floating elements .....130

CHAPTER 3: **HTML5 Forms** .....135
Understanding HTML5 Forms .....135
Defining a form .....136
Working with form fields .....137
Using Input Fields .....138
Text boxes .....138
Password entry .....140
Check boxes .....141
Radio buttons .....142
Hidden fields .....143
File upload .....144
Buttons .....145
Adding a Text Area .....146
Using Drop-Down Lists .....147
Enhancing HTML5 Forms .....149
Data lists .....149
Additional input fields .....150
Using HTML5 Data Validation .....154
Holding your place .....154
Making certain data required .....155
Validating data types .....155

CHAPTER 4: **Advanced CSS3** .....157
Rounding Your Corners .....157
Using Border Images .....159
Looking at the CSS3 Colors .....162
Playing with Color Gradients .....164
Linear gradients .....164
Radial gradients .....165
Adding Shadows .....166
Text shadows .....166
Box shadows .....167
Creating Fonts .....168
Focusing on font files .....169
Working with web fonts .....169
Handling Media Queries .....171
Using the @media command .....171
Dealing with CSS3 media queries .....172
Applying multiple style sheets .....175

CHAPTER 5: **HTML5 and Multimedia** . . . . . 177
Working with Images . . . . . 177
Placing images . . . . . 178
Styling images . . . . . 179
Linking images . . . . . 181
Working with image maps . . . . . 182
Using HTML5 image additions . . . . . 183
Playing Audio . . . . . 185
Embedded audio . . . . . 185
Digital audio formats . . . . . 186
Audio the HTML5 way . . . . . 188
Watching Videos . . . . . 190
Paying attention to video quality . . . . . 190
Looking at digital video formats . . . . . 191
Putting videos in your web page . . . . . 192
Getting Help from Streamers . . . . . 194

**BOOK 3: JAVASCRIPT** . . . . . 195

CHAPTER 1: **Introducing JavaScript** . . . . . 197
Knowing Why You Should Use JavaScript . . . . . 197
Changing web page content . . . . . 198
Changing web page styles . . . . . 198
Seeing Where to Put Your JavaScript Code . . . . . 199
Embedding JavaScript . . . . . 199
Using external JavaScript files . . . . . 203
The Basics of JavaScript . . . . . 203
Working with data . . . . . 204
Data types . . . . . 205
Arrays of data . . . . . 206
Operators . . . . . 207
Controlling Program Flow . . . . . 209
Conditional statements . . . . . 209
Loops . . . . . 216
Working with Functions . . . . . 220
Creating a function . . . . . 221
Using a function . . . . . 222

CHAPTER 2: **Advanced JavaScript Coding** . . . . . 223
Understanding the Document Object Model . . . . . 223
The Document Object Model tree . . . . . 224
JavaScript and the Document Object Model . . . . . 226

Finding Your Elements . . . . . 233
Getting to the point . . . . . 233
Walking the tree . . . . . 235
Working with Document Object Model Form Data . . . . . 238
Text boxes . . . . . 238
Text areas . . . . . 239
Check boxes . . . . . 240
Radio buttons . . . . . 241

CHAPTER 3: **Using jQuery** . . . . . 243
Loading the jQuery Library . . . . . 244
Option 1: Downloading the library file to your server . . . . . 245
Option 2: Using a content delivery network . . . . . 246
Using jQuery Functions . . . . . 246
Finding Elements . . . . . 247
Replacing Data . . . . . 250
Working with text . . . . . 250
Working with HTML . . . . . 252
Working with attributes . . . . . 253
Working with form values . . . . . 253
Changing Styles . . . . . 254
Playing with properties . . . . . 254
Using CSS objects . . . . . 256
Using CSS classes . . . . . 257
Changing the Document Object Model . . . . . 259
Adding a node . . . . . 259
Removing a node . . . . . 260
Playing with Animation . . . . . 261

CHAPTER 4: **Reacting to Events with JavaScript and jQuery** . . . 263
Understanding Events . . . . . 263
Event-driven programming . . . . . 264
Watching the mouse . . . . . 264
Listening for keystrokes . . . . . 265
Paying attention to the page itself . . . . . 266
Focusing on JavaScript and Events . . . . . 267
Saying hello and goodbye . . . . . 267
Listening for mouse events . . . . . 269
Listening for keystrokes . . . . . 273
Event listeners . . . . . 275
Looking at jQuery and Events . . . . . 276
jQuery event functions . . . . . 276
The jQuery event handler . . . . . 280

CHAPTER 5: **Troubleshooting JavaScript Programs** . . . 283
Identifying Errors . . . 283
Working with Browser Developer Tools . . . 285
The DOM Explorer . . . 286
The Console . . . 287
The Debugger . . . 290
Working Around Errors . . . 295

**BOOK 4: PHP** . . . 301

CHAPTER 1: **Understanding PHP Basics** . . . 303
Seeing the Benefits of PHP . . . 303
A centralized programming language . . . 304
Centralized data management . . . 304
Understanding How to Use PHP . . . 305
Embedding PHP code . . . 305
Identifying PHP pages . . . 306
Displaying output . . . 307
Handling new-line characters . . . 309
Working with PHP Variables . . . 310
Declaring variables . . . 311
Seeing which data types PHP supports . . . 312
Grouping data values with array variables . . . 315
Using PHP Operators . . . 317
Arithmetic operators . . . 317
Arithmetic shortcuts . . . 318
Boolean operators . . . 319
String operators . . . 320
Including Files . . . 320
The include() function . . . 320
The require() function . . . 323

CHAPTER 2: **PHP Flow Control** . . . 325
Using Logic Control . . . 325
The if statement . . . 326
The else statement . . . 328
The elseif statement . . . 328
The switch statement . . . 330
Looping . . . 331
The while family . . . 331
The for statement . . . 333
The foreach statement . . . 334
Building Your Own Functions . . . 336
Working with Event-Driven PHP . . . 339

Working with links . . . . . . . . . . 339
Processing form data . . . . . . . . . . 343

CHAPTER 3: **PHP Libraries** . . . . . . . . . . 349
How PHP Uses Libraries . . . . . . . . . . 349
Exploring PHP extensions . . . . . . . . . . 350
Examining the PHP extensions . . . . . . . . . . 351
Including extensions . . . . . . . . . . 353
Adding additional extensions . . . . . . . . . . 354
Text Functions . . . . . . . . . . 354
Altering string values. . . . . . . . . . . 354
Splitting strings . . . . . . . . . . 356
Testing string values . . . . . . . . . . 359
Searching strings . . . . . . . . . . 360
Math Functions . . . . . . . . . . 361
Number theory . . . . . . . . . . 361
Calculating logs and exponents. . . . . . . . . . . 362
Working the angles . . . . . . . . . . 363
Hyperbolic functions . . . . . . . . . . 364
Tracking statistics . . . . . . . . . . 364
Date and Time Functions . . . . . . . . . . 365
Generating dates . . . . . . . . . . 365
Using timestamps . . . . . . . . . . 367
Calculating dates . . . . . . . . . . 368
Image-Handling Functions . . . . . . . . . . 369

CHAPTER 4: **Considering PHP Security** . . . . . . . . . . 375
Exploring PHP Vulnerabilities . . . . . . . . . . 375
Cross-site scripting . . . . . . . . . . 376
Data spoofing . . . . . . . . . . 379
Invalid data . . . . . . . . . . 380
Unauthorized file access. . . . . . . . . . . 382
PHP Vulnerability Solutions . . . . . . . . . . 384
Sanitizing data . . . . . . . . . . 384
Validating data . . . . . . . . . . 389

CHAPTER 5: **Object-Oriented PHP Programming** . . . . . . . . . . 395
Understanding the Basics of Object-Oriented Programming . . . . . . 395
Defining a class . . . . . . . . . . 396
Creating an object instance . . . . . . . . . . 397
Using Magic Class Methods . . . . . . . . . . 401
Defining mutator magic methods . . . . . . . . . . 401
Defining accessor magic methods . . . . . . . . . . 403
The constructor . . . . . . . . . . 406

The destructor .407
Copying objects .408
Displaying objects .408
Loading Classes .409
Extending Classes .414

CHAPTER 6: **Sessions and Carts** .419
Storing Persistent Data .419
The purpose of HTTP cookies .420
Types of cookies .421
The anatomy of a cookie .422
Cookie rules .424
PHP and Cookies .424
Setting cookies .424
Reading cookies .426
Modifying and deleting cookies .428
PHP and Sessions .430
Starting a session .431
Storing and retrieving session data .431
Removing session data .435
Shopping Carts .436
Creating a cart .436
Placing items in the cart .437
Retrieving items from a cart .437
Removing items from a cart .438
Putting it all together .438

**BOOK 5: MYSQL** .443

CHAPTER 1: **Introducing MySQL** .445
Seeing the Purpose of a Database .445
How databases work .446
Relational databases .449
Database data types .451
Data constraints .451
Structured Query Language .452
Presenting MySQL .454
MySQL features .454
Storage engines .456
Data permissions .457
Advanced MySQL Features .458
Handling transactions .458
Making sure your database is ACID compliant .459
Examining the views .461

Working with stored procedures. . . . . . . . . . . . . . . . . . . . . . . . . .462
Pulling triggers . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .463
Working with blobs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .463

CHAPTER 2: **Administering MySQL**. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 465
MySQL Administration Tools . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .465
Working from the command line . . . . . . . . . . . . . . . . . . . . . . . . . .466
Using MySQL Workbench. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .470
Using the phpMyAdmin tool . . . . . . . . . . . . . . . . . . . . . . . . . . . . .475
Managing User Accounts . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .477
Creating a user account . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .477
Managing user privileges . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .481

CHAPTER 3: **Designing and Building a Database** . . . . . . . . . . . . . . . . 489
Managing Your Data . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .489
The first normal form . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .490
The second normal form . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .491
The third normal form . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .491
Creating Databases . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .492
Using the MySQL command line . . . . . . . . . . . . . . . . . . . . . . . . . .492
Using MySQL Workbench. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .495
Using phpMyAdmin. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .497
Building Tables. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .500
Working with tables using the command-line interface . . . . . . . .500
Working with tables using Workbench. . . . . . . . . . . . . . . . . . . . . .505
Working with tables in phpMyAdmin . . . . . . . . . . . . . . . . . . . . . .508

CHAPTER 4: **Using the Database** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 513
Working with Data. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .513
The MySQL command-line interface . . . . . . . . . . . . . . . . . . . . . . .514
The MySQL Workbench tool . . . . . . . . . . . . . . . . . . . . . . . . . . . . .519
The phpMyAdmin tool . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .522
Searching for Data. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .524
The basic SELECT format . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .525
More advanced queries . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .527
Playing It Safe with Data . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .531
Performing data backups. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .532
Restoring your data. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .538

CHAPTER 5: **Communicating with the Database from PHP Scripts** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 541
Database Support in PHP. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .541
Using the mysqli Library. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .543
Connecting to the database. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .544
Closing the connection . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .545

Submitting queries . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .546
Retrieving data. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .547
Being prepared . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .549
Checking for errors . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .551
Miscellaneous functions. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .553
Putting It All Together. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .554

## BOOK 6: CREATING OBJECT-ORIENTED PROGRAMS. . . . . .561

CHAPTER 1: **Designing an Object-Oriented Application**. . . . . . . . .563
Determining Application Requirements. . . . . . . . . . . . . . . . . . . . . . . . . .563
Creating the Application Database . . . . . . . . . . . . . . . . . . . . . . . . . . . .565
Designing the database . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .565
Creating the database. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .568
Designing the Application Objects . . . . . . . . . . . . . . . . . . . . . . . . . . . . .571
Designing objects . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .571
Coding the objects in PHP . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .573
Designing the Application Layout . . . . . . . . . . . . . . . . . . . . . . . . . . . . .579
Designing web page layout . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .580
The AuctionHelper page layout. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .581
Coding the Website Layout . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .582
Creating the web page template. . . . . . . . . . . . . . . . . . . . . . . . . . . . .582
Creating the support files. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .587

CHAPTER 2: **Implementing an Object-Oriented Application** . . .593
Working with Events . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .593
Bidder Object Events. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .595
Listing bidders . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .595
Adding a new bidder. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .603
Searching for a bidder . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .605
Item Object Events . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .605
Listing items . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .606
Adding a new item. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .611
Searching for an item . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .614
Logging Out of a Web Application. . . . . . . . . . . . . . . . . . . . . . . . . . . . .614
Testing Web Applications. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .616

CHAPTER 3: **Using AJAX** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .619
Getting to Know AJAX . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .619
Communicating Using JavaScript . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .621
Considering XMLHttpRequest class methods . . . . . . . . . . . . . . . . . .622
Focusing on XMLHttpRequest class properties. . . . . . . . . . . . . . . . .623
Trying out AJAX. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .625

Using the jQuery AJAX Library . . . . . . . . . . 629
The jQuery $.ajax() function . . . . . . . . . . 629
The jQuery $.get() function . . . . . . . . . . 633
Transferring Data in AJAX . . . . . . . . . . 635
Looking at the XML standard . . . . . . . . . . 635
Using XML in PHP . . . . . . . . . . 636
Using XML in JavaScript . . . . . . . . . . 640
Modifying the AuctionHelper Application . . . . . . . . . . 643

CHAPTER 4: **Extending WordPress** . . . . . . . . . . 651
Getting Acquainted with WordPress . . . . . . . . . . 651
What WordPress can do for you . . . . . . . . . . 652
How to run WordPress . . . . . . . . . . 653
Parts of a WordPress website . . . . . . . . . . 654
Installing WordPress . . . . . . . . . . 655
Downloading the WordPress software . . . . . . . . . . 655
Creating the database objects . . . . . . . . . . 656
Configuring WordPress . . . . . . . . . . 658
Examining the Dashboard . . . . . . . . . . 662
Using WordPress . . . . . . . . . . 664
Exploring the World of Plugins . . . . . . . . . . 669
WordPress APIs . . . . . . . . . . 670
Working with plugins and widgets . . . . . . . . . . 671
Creating Your Own Widget . . . . . . . . . . 674
Coding the widget . . . . . . . . . . 674
Activating the widget plugin . . . . . . . . . . 676
Adding the widget . . . . . . . . . . 677

**BOOK 7: USING PHP FRAMEWORKS** . . . . . . . . . . 681

CHAPTER 1: **The MVC Method** . . . . . . . . . . 683
Getting Acquainted with MVC . . . . . . . . . . 683
Exploring the MVC method . . . . . . . . . . 684
Digging into the MVC components . . . . . . . . . . 686
Communicating in MVC . . . . . . . . . . 690
Comparing MVC to Other Web Models . . . . . . . . . . 691
The MVP method . . . . . . . . . . 692
The MVVM method . . . . . . . . . . 692
Seeing How MVC Fits into N-Tier Theory . . . . . . . . . . 693
Implementing MVC . . . . . . . . . . 694

CHAPTER 2: **Selecting a Framework** . . . . . 695
Getting to Know PHP Frameworks . . . . . 695
Convention over configuration . . . . . 696
Scaffolding . . . . . 698
Routing . . . . . 699
Helper methods . . . . . 700
Form validation . . . . . 700
Support for mobile devices . . . . . 700
Templates . . . . . 701
Unit testing . . . . . 701
Knowing Why You Should Use a Framework . . . . . 702
Focusing on Popular PHP Frameworks . . . . . 704
CakePHP . . . . . 704
CodeIgniter . . . . . 705
Laravel . . . . . 707
Symfony . . . . . 708
Zend Framework . . . . . 709
Looking At Micro Frameworks . . . . . 710
Lumen . . . . . 710
Slim . . . . . 711
Yii . . . . . 713

CHAPTER 3: **Creating an Application Using Frameworks** . . . . . 715
Building the Template . . . . . 715
Initializing the application . . . . . 716
Exploring the files and folders . . . . . 718
Defining the database environment . . . . . 719
Creating an Application Scaffold . . . . . 721
Installing the scaffolding . . . . . 721
Exploring the scaffolding code . . . . . 724
Modifying the Application Scaffold . . . . . 725
Adding a new feature link . . . . . 726
Creating the controller code . . . . . 728
Modifying the model code . . . . . 730
Painting a view . . . . . 731

INDEX . . . . . 735

# Introduction

The Internet has become an amazing place to shop, do your banking, look up homework assignments, and even keep track of your bowling league scores. Behind all those great applications are a bunch of different web technologies that must all work together to create the web experience you come to expect.

You may think that creating web applications is best left for the professionals, but you'd be surprised by just how well you can do with just a little knowledge and experience! That's the point of this book.

## About This Book

Think of this book as a reference book. Like the dictionary or an encyclopedia (remember those?), you don't have to read it from beginning to end. Instead, you can dip into the book to find the information you need and return to it again when you need more. That said, you won't be disappointed if you work through the book from beginning to end, and you may find it easier to follow along with some of the examples.

In this book, I walk you through all the different technologies involved with creating dynamic web applications that can track data and present it in an orderly and pleasing manner. I cover several key topics that you'll need to know to create a full-featured, dynamic web application:

» **Creating the basic layout of a web page:** In this book, you see the program code behind placing content on a web page and reacting to your website visitors' mouse clicks.

» **Styling the web page:** Just placing data on a web page is boring. In this book, you learn how to use CSS to help use color, images, and placement to help liven up your web applications.

» **Adding dynamic features:** These days, having a static web page that just sits there doesn't get you many followers. This book shows you how to incorporate JavaScript to animate your web pages and provide dynamic features.

- **Leveraging the power of the server:** The PHP programming language allows you to harness the power behind the web server to dynamically generate web pages "on the fly" as your website visitors make choices.
- **Storing data for the future:** Just about every dynamic web application needs to store data, and in this book you learn exactly how to do that using the MySQL server, which is commonly available in just about every web platform.
- **Creating full applications:** Many books throw a bunch of technology at you and expect you to put the pieces together yourself. This book not only shows you the technology, but also demonstrates how all the parts fit together to create a dynamic web application.
- **Using helper programs:** No one is an island; everyone needs some help putting together those fancy web applications. There are plenty of tools to help you get the job done, and with this book you find out which tools will help you with which features of your application.

Throughout this book you see sidebars (text in gray boxes) and material marked with the Technical Stuff icon. All of these things are skippable. If you have time and are interested, by all means read them, but if you don't or aren't, don't.

Finally, within this book, you may note that some web addresses break across two lines of text. If you're reading this book in print and want to visit one of these web pages, simply key in the web address exactly as it's noted in the text, pretending as though the line break doesn't exist. If you're reading this as an e-book, you've got it easy — just click the web address to be taken directly to the web page.

## Foolish Assumptions

You don't need any level of programming experience to enjoy this book and start creating your own web applications. Each chapter walks through all the basics you need to know and doesn't assume you've ever coded before. As long as you're reasonably comfortable navigating your way around a standard desktop computer, you have all the experience you need!

That said, if you've already tried your hand at web programming and you just want to fill in a few holes, this book will work well for you, too!

This book doesn't expect you to run out and buy any expensive software packages to start your web development career. All the tools that are used in the book are freely available open-source software. I walk you through how to set up a complete development environment, whether you're working in Microsoft Windows, Apple macOS, or Linux.

## Icons Used in This Book

I use some icons throughout the book to help you identify useful information. Here's what the icons are and what I use them for:

TIP

Anything marked with the Tip icon provides some additional information about a topic to help you better understand what's going on behind the scenes or how to better use the feature discussed in the text.

REMEMBER

You don't have to commit this book to memory — there won't be a test. But every once in a while I tell you something so important that you should remember it. When I do, I mark it with the Remember icon.

WARNING

The Warning icon is there to point out potential pitfalls that can cause problems. If you want to save yourself a lot of time or trouble, heed these warnings.

TECHNICAL STUFF

When you see the Technical Stuff icon, be prepared to put your geek hat on. When I get into the weeds, I use the Technical Stuff icon. If you're not interested in these details, feel free to skip these sections — you won't miss anything essential about the topic at hand.

## Beyond the Book

In addition to the material in the print or e-book you're reading right now, you also get access to a free online Cheat Sheet filled with more tips and tricks on building a web application, including accessing any database from your PHP programs, filtering data your program receives from web forms to block unwanted or potentially dangerous data, quickly finding data in a MySQL database, and triggering JavaScript events at predetermined times in a browser. To access this resource go to `www.dummies.com` and enter **PHP, MySQL & JavaScript All-in-One For Dummies Cheat Sheet** in the search box.

## Where to Go from Here

This book doesn't have to be read from beginning to end, so you can dive in wherever you want! Use the Table of Contents and Index to find subjects that interest you. If you already know PHP and JavaScript and you're just interested in learning how to create a dynamic web application from scratch, start out with

Book 6, Chapter 1. If you're interested in learning how to use one of the framework packages available for PHP, check out Book 7, Chapter 1. Or, if you're interested in everything, start with Book 1, Chapter 1, and read until the very end.

With the information in this book, you'll be ready to start creating your own dynamic web applications. Web programming is one of those skills that takes time and practice to get good at, so the more coding you can do, the better you'll get at it. To get some practice, you may want to offer your services for free at first, to build up a reputation. Find a needy nonprofit organization that you're interested in supporting and offer to work on its website. They'll get a great website, and you'll get a project to add to your résumé!

Don't stop learning! There are always new things coming out in the web world, even if you just stick to using the same software packages to develop your web applications. Stay plugged in to the PHP world by visiting the official PHP website at `www.php.net` or by visiting (and even participating in) one or more of the many PHP forums. Just do some Googling to find them.

Enjoy your newfound skills in developing dynamic web applications!

# 1 Getting Started with Web Programming

# Contents at a Glance

CHAPTER 1: **Examining the Pieces of Web Programming** . . . . . . 7
Creating a Simple Web Page . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
Creating a Dynamic Web Page . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
Storing Content . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25

CHAPTER 2: **Using a Web Server** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
Recognizing What's Required. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
Considering Your Server Options . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
Tweaking the Servers . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 41

CHAPTER 3: **Building a Development Environment** . . . . . . . . . . . 51
Knowing Which Tools to Avoid . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
Working with the Right Tools. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53

**IN THIS CHAPTER**

» **Understanding how simple web pages work**

» **Incorporating programming into your web page**

» **Storing content in a database**

Chapter 1

# Examining the Pieces of Web Programming

At first, diving into web programming can be somewhat overwhelming. You need to know all kinds of things in order to build a web application that not only looks enticing but also works correctly. The trick to learning web programming is to pull the individual pieces apart and tackle them one at a time.

This chapter gets you started on your web design journey by examining the different pieces involved in creating a simple web page. Then it kicks things up a notch and walks you through dynamic web pages. And finally, the chapter ends by explaining how to store your content for use on the web.

## Creating a Simple Web Page

Before you can run a marathon, you need to learn how to walk. Likewise, before you can create a fancy website, you need to know the basics of how web pages work.

Nowadays, sharing documents on the Internet is easy, but it wasn't always that way. Back in the early days of the Internet, documents were often created using proprietary word-processing packages and had to be downloaded using the cumbersome File Transfer Protocol (FTP). To retrieve a document, you had to know exactly what server contained the document, you had to know where it was stored on the server, and you had to be able to log into the server. After all that, you *still* needed to have the correct word-processing software on your computer to view the document. As you can imagine, it wasn't long before a new way of sharing content was required.

To get to where we are today, several different technologies had to be developed:

- A method for linking related documents together
- A way for the document reader to display formatted text the same way in any type of device
- An Internet standard allowing clients to easily retrieve documents from any server
- A standard method of styling and positioning content in documents

This section describes the technology that made viewing documents on the Internet work the way it does today.

## Kicking things off with the World Wide Web

In 1989, Tim Berners-Lee developed a method of interconnecting documents to make sharing research information on the Internet easier. His creation, the *World Wide Web,* defined a method for linking documents together in a web structure, so that a researcher could follow the path between related documents, no matter where they were located in the world. Clicking text in one document took you to another document automatically, without your having to manually find and download the related document.

The method Berners-Lee developed for linking documents is called *hypertext.* Hypertext embeds links that are hidden from view in the document, and directs the software being used to view the document (known as the *web browser*) to retrieve the referenced document. With hypertext, you just click the link, and the software (the web browser) does all the work of finding and retrieving the related document for you.

Because the document-viewing software does all the hard work, a new type of software had to be developed that was more than just a document viewer. That's where web browsers came into existence. Web browsers display a document on a computer screen and respond to the reader clicking hypertext links to retrieve other specified documents.

To implement hypertext in documents, Berners-Lee had to utilize a text-based document-formatting system. Fortunately for him, a lot of work had already been done on that.

## Making sense of markup languages

*Markup languages* were developed to replace proprietary word-processing packages with a standard way of formatting documents so that they could be read by any type of document viewer on any type of device. This goal is accomplished by embedding *tags* in the text. Each tag indicates a formatting feature, such as headings, bold or italic text, or special margins. What made markup languages different from word-processing packages is that these tags were common text codes instead of proprietary codes, making it generic enough that any device could read and process them.

The first popular markup language was the Generalized Markup Language (GML), developed by IBM in the 1960s. The International Organization for Standardization (ISO) took up the challenge of creating markup languages and produced the Standard Generalized Markup Language (SGML), mainly based on GML, in the 1980s. However, because SGML was developed to cover all types of document formatting on all types of devices, it's extremely complex and it wasn't readily adapted.

Berners-Lee used the ideas developed in SGML to create a simplified markup language that could support his hypertext idea. He called it *Hypertext Markup Language* (HTML). HTML uses the same concept of tags that SGML uses, but it defines fewer of them, making it easier to implement in software.

An example of an HTML tag is `<h1>`. You use this tag to define text that's used as a page heading. Just surround the text with an opening `<h1>` tag, and a corresponding closing `</h1>` tag, like this:

```
<h1>This is my heading</h1>
```

When the browser gets to the `<h1>` tag, it knows to format the text embedded in the opening and closing tags using a different style of formatting, such as a larger font or a bold typeface.

To define a hypertext link to another document, you use the `<a>` tag:

```
<a href="anotherdoc.html">Click here for more info</a>
```

When the reader clicks the *Click here for more info* text, the browser automatically tries to retrieve the document specified in the `<a>` tag. That document can be on the same server or on another server anywhere on the Internet.

HTML development has seen quite a few changes since Berners-Lee created it and turned it over to the World Wide Web Consortium (W3C) to maintain. Table 1-1 shows the path the language has taken.

**TABLE 1-1** HTML Versions

| Version | Description |
|---|---|
| HTML 1.0 | Formally released in 1989 as the first public version of HTML |
| HTML 2.0 | Released in 1995 to add interactive elements |
| HTML 3.0 | Released in 1996 but never widely adopted |
| HTML 3.2 | Released in 1997, adding support for tables |
| HTML 4.01 | Released in 1999, widely adopted, and remains an often-used standard |
| XHTML 1.0 | Released in 2001, standardizing HTML around the XML document format |
| XHTML 1.1 | Released in 2002, making updates and corrections to XHTML 1.1 |
| HTML 5.0 | Released in 2014, adding multimedia features |

The HTML version 4.01 standard was the backbone of websites for many years, and it's still used by many websites today. However, HTML version 5.0 (called HTML5 for short) is the future of web development. It provides additional features for embedding multimedia content in web pages without the need for proprietary software plug-ins (such as Adobe Flash Player). Because multimedia is taking over the world (just ask YouTube), HTML5 has grown in popularity. This book focuses on HTML5; all the code included in this book use that standard.

## Retrieving HTML documents

Besides a document-formatting standard, Berners-Lee also developed a method of easily retrieving the HTML documents in a client–server environment. A *web server* software package runs in the background on a server, listening for connection requests from *web clients* (the browser). The browser sends requests to retrieve HTML documents from the server. The request can be sent anonymously (without using a login username), or the browser can send a username and password or certificate to identify the requestor.

These requests and responses are defined in the *Hypertext Transfer Protocol* (HTTP) standard. HTTP defines a set of requests the client can send to the server and a set of responses the server uses to reply back to the client.

This section walks you through the basics of how web servers and web clients use HTTP to interact with each other to move web pages across the Internet.

### Web clients

The web client sends requests to the web server on a standard network communication channel (known as TCP port 80), which is defined as the standard for

HTTP communication. HTTP uses standard text requests sent to the server, either requesting information from the server or sending information to the server. Table 1-2 shows the basic HTTP client requests available.

**TABLE 1-2** HTTP Client Requests

| Request | Description |
|---|---|
| `CONNECT` | Converts the connection into a secure tunnel for sending data |
| `DELETE` | Deletes the specified resource |
| `GET` | Requests the specified resource |
| `HEAD` | Requests the title of the specified resource |
| `OPTIONS` | Retrieves the HTTP requests that the server supports |
| `PATCH` | Applies a modification to a resource |
| `POST` | Sends specified data to the server for processing |
| `PUT` | Stores specified data at a specified location |
| `TRACE` | Sends the received request back to the client |

As shown in Table 1-2, when you ask to view a web page from your client browser, the browser sends the HTTP `GET` request to the server, specifying the filename of the web page. The server then responds with a response code along with the requested data. If the client doesn't specify a filename in the `GET` request, most servers have a default file with which to respond.

## Web servers

With HTTP, the web server must respond to each client request received. If the client sends a request that the server can't process, the server must send some type of error code back to the client indicating that something went wrong.

The first part of the server response is a status code and text that the client uses to determine whether the submitted request was successful. The format of the HTTP response uses a three-digit status code, followed by an optional text message that the browser can display. The three-digit codes are broken down into five categories:

- **1xx:** Informational messages
- **2xx:** Success
- **3xx:** Redirection

- **4xx:** Client error
- **5xx:** Server error

The three-digit status code is crucial to knowing what happened with the response. Many status codes are defined in the HTTP standards, providing some basic information on the status of client requests. Table 1-3 shows just a few of the standard HTTP response codes that you may run into.

**TABLE 1-3** Common HTTP Server Response Status Codes

| Status Code | Text Message | Description |
|---|---|---|
| 100 | Continue | The client should send additional information. |
| 101 | Switching Protocols | The server is using a different protocol for the request. |
| 102 | Processing | The server is working on the response. |
| 200 | OK | The server accepted the request and has returned the response. |
| 201 | Created | The server created a new resource in response to the request. |
| 202 | Accepted | The data sent by the client has been accepted by the server but has not completed processing the data. |
| 206 | Partial Content | The response returned by the server is only part of the full data; more will come in another response. |
| 300 | Multiple Choices | The request matched multiple possible responses from the server. |
| 301 | Moved Permanently | The requested file was moved and is no longer at the requested location. |
| 302 | Found | The requested resource was found at a different location. |
| 303 | See Other | The requested resource is available at a different location. |
| 304 | Not Modified | The requested resource was not modified since the last time the client accessed it. |
| 307 | Temporary Redirect | The requested resource was temporarily moved to a different location. |
| 308 | Permanent Redirect | The requested resource was permanently moved to a different location. |
| 400 | Bad Request | The server cannot process the request. |
| 401 | Unauthorized | The resource requires authentication that the client did not provide. |

| Status Code | Text Message | Description |
|---|---|---|
| 402 | Payment Required | The requested resource is not freely available. |
| 403 | Forbidden | The resource requires authentication, and the client does not have the proper permission. |
| 404 | Not Found | The requested resource was not located on the server. |
| 414 | URI Too Long | The Uniform Resource Identifier (URI) describing the location of the resource was longer than the server is able to handle. |
| 415 | Unsupported Media Type | The server does not know how to process the requested resource file. |
| 429 | Too Many Requests | The client has sent too many requests within a specific amount of time. |
| 500 | Internal Server Error | An unexpected condition occurred on the server while trying to retrieve the requested resource. |
| 501 | Not Implemented | The server doesn't recognize the request. |
| 502 | Bad Gateway | The server was acting as a proxy to another server but received an invalid response from the other server. |
| 503 | Service Unavailable | The server is currently unavailable, often due to maintenance. |
| 505 | HTTP Version Not Supported | The server doesn't support the HTTP standard used by the client in the request. |
| 507 | Insufficient Storage | The server is unable to store the resource due to lack of storage space. |
| 511 | Network Authentication Required | The client is required to authenticate with a network resource to receive the response. |

As you can see from Table 1-3, a web server can return many possible responses. It's the client's job to parse the response and determine the next action to take.

If the response indicates the request was successful, the server will follow the response code with the data related to the request, such as the contents of an HTML file. The client must then read the returned data and decide what to do with it. For HTML files, the browser will display the requested file, applying the HTML formatting tags to the data.

TIP

Don't worry about trying to memorize all the HTTP status codes. Most of them you'll never run into in your web-programming career. Before long, you'll start to remember a few of the more common ones, and you can always look up any others you run into.

## Styling

The HTML standard defines how browsers perform basic formatting of text, but it doesn't really provide a way to tell a browser how to display the text. The `<h1>` tag indicates that the text should be a heading, but nothing tells the browser just how to display the heading to make it different from any other text on the page.

This is where styling comes into play. *Styling* allows you to tell the browser just what fonts, sizes, and colors to use for text, as well as how to position the text in the display. This section explains how styling affects how your web pages appear to your visitors.

### Style sheets

There are several ways to define styling for an HTML document. The most basic method is what the browser uses by default. When the browser sees an HTML formatting tag, such as the `<h1>` tag, it has a predefined font, size, and color that the developer of the browser felt was useful.

That's fine, but what if you want to make some headings black and others red? This is possible with *inline styling.* Inline styling allows you to define special styles that apply to only one specific tag in the document. For example, to make one heading red, you'd use the following HTML:

```
<h1 style="color: red">Warning, this is bad</h1>
```

The `style` term is called an *attribute* of the `<h1>` tag. There are a few different attributes you can apply directly to tags within HTML; each one modifies how the browser should handle the tag. The `style` attribute allows you to apply any type of styling to this specific `<h1>` tag in the document. In this example, I chose to change the color of the text.

Now, you're probably thinking that I've just opened another can of worms. What if you want to apply the red color to *all* the <h1> tags in your document? That's a lot of extra code to write! Don't worry, there's a solution for that.

Instead of inserting styles inline, you can create a style definition that applies to the entire document. This method is known as *internal styling.* It defines a set of styles at the top of the HTML document that are applied to the entire document. Internal styling looks like this:

```
<style>
h1 {color: red;}
</style>
```

Now the browser will display all the `<h1>` tags in the document using a red color. But wait, there's more!

Style listings can be somewhat lengthy for large web pages, and placing them at the top of a document can become cumbersome. Also, if you want to apply the same styles to all the web pages in a website, having to retype or copy all that text can be tiring. To solve that problem, you use an external style sheet.

An *external style sheet* allows you to define styles just as the internal method does, but in a separate file, called a *style sheet.* Any web page can reference the same style sheet, and you can apply multiple style sheets to a single web page. You reference the external style sheet using the `<link>` tag, like this:

```
<link rel="stylesheet" href="mystyles.css">
```

When the browser sees this tag, it downloads the external style sheet, and applies the styles you defined in it to the document.

This all sounds great, but things just got a lot more complicated! Now there are three different locations from which you can define styles for your HTML document, on top of what the browser itself does. How are you supposed to know which ones take precedence over the others?

The *Cascading Style Sheet* (CSS) standard defines a set of rules that determine just how browsers should apply styles to an HTML document. As the name implies, styles cascade down from a high level to a low level. Styles defined in a higher-level rule override styles defined in a lower-level rule.

The CSS standard defines nine separate levels, which I cover in greater detail in Book 2, Chapter 2, but for now, here are the four most common style levels, in order from highest priority to lowest:

- Styles defined within the element tags
- Styles defined in an internal style sheet
- Styles defined in an external style sheet
- Styles defined by the client's browser defaults

So, any style attributes you set in an element tag override any styles that you set in an internal style sheet, which overrides any styles you set in an external style sheet, which overrides any styles the client browser uses by default. This allows you to set an overall style for your web pages using an external style sheet, and then override those settings for individual situations using the standard element tags.

TECHNICAL STUFF

You may be wondering how assistive technology tools work to change the web page display for individuals who are sight impaired. Part of the nine rules that I cover in Book 2, Chapter 2, incorporate any rules defined in the browser for sight-impaired viewing.

## CSS standards

The CSS standard defines a core set of styles for basic rendering of an HTML document. The first version of CSS (called CSS1) was released in 1996, and it only defined some very rudimentary styles:

- Font type, size, and color
- Text alignment (such as margins)
- Background colors or images
- Borders

The second version of CSS, called — you guessed it! — CSS2, was released in 1998. It added only a few more styling features:

- More-exact positioning of text
- Styles for different output types (such as printers or screens)
- The appearance of browser features such as the cursor and scrollbar

That's still not all that impressive of a list of styles. Needless to say, more was needed to help liven up web pages. To compensate for that, many browser developers started creating their own style definitions, apart from the CSS standards. These style definitions are called *extensions.* The browser extensions covered lots of different fancy styling features, such as applying rounded edges to borders and images, making a smoother layout in the web page.

As you might guess, having different extensions to apply different style features in different browsers just made things more complicated. Instead of coding a single style for an element in an HTML document, you needed to code the same feature several different ways so the web page would look the same in different browsers. This quickly became a nightmare.

When work was started on the CSS3 standard in 1999, one of the topics was to rein in the myriad browser extensions. However, things quickly became complicated because all the different browser developers wanted their own extensions included in the new standard.

To simplify the process, the CSS design committee split the CSS standards into separate modules. Each CSS module covers a specific area of styling, such as colors, media support, and backgrounds. Each module could be voted on and released under a different timeline. The downside to this approach is that now each module has been released as a recommended standard at a different time, making the CSS3 standard somewhat difficult to track and implement.

Quite possibly one of the most anticipated features of CSS3 is the ability to define fonts. Fonts have long been the bane of web programmers. When you define a specific font, that font must be installed on your website visitor's computer in order for the browser to use it. If the font isn't available, the browser picks a default font to use, which often becomes an ugly mess.

*Web fonts* allow you to define a font on your server so that every client browser can download the font and render text using it. This is a huge accomplishment! No longer are you reliant on your website visitors having specific fonts installed in their web browsers.

Yet another popular feature of CSS3 is the use of shadows and semitransparent colors in text and other web page elements, such as form objects. These features by themselves can transform an ugly HTML form into a masterpiece.

The combination of HTML5 and CSS3 has greatly revolutionized the web world, allowing developers to create some pretty amazing websites. However, one thing was still missing: the ability to easily change content on the web page.

## Creating a Dynamic Web Page

*Static web pages* contain information that doesn't change until the web designer or programmer manually changes it. In the early days of the Internet, simply jumping on the Internet bandwagon was important for corporations. It wasn't so important what companies posted on the web, as long as they had an Internet presence where customers could get basic information about the company and its products. Static web pages, consisting solely of HTML and CSS, easily accomplished this function.

But one of the big limitations of static web pages is how much effort it takes to update them. Changing a single element on a static web page requires rebuilding and reloading the entire page, or sometimes even a group of web pages. This process is way too cumbersome for an organization that frequently needs to post real-time information, such as events, awards, or closings. Also, during this process, a developer can accidentally change other items on the page, seriously messing up the information on the web page, or even the entire web page layout!

*Dynamic web pages* allow you to easily change your content in real time without even touching the coding of the page. That's right: Without manually making any changes to the page itself, the information on the page can change. This means you can keep the content on the page fresh so that what a visitor sees there now may be updated or replaced in a day, an hour, or a minute. The core layout of the web page can remain the same, but the data presented constantly changes.

To successfully create a dynamic web page, you have to know a method for automatically inserting real-time data into the HTML code that gets sent to the client browser. This is where web scripting languages come in.

A *web scripting language* allows you to insert program code inside your web page that dynamically generates HTML that the client browser reads. A processor reads the program code and dynamically generates HTML to display content on the web page, as shown in Figure 1-1.

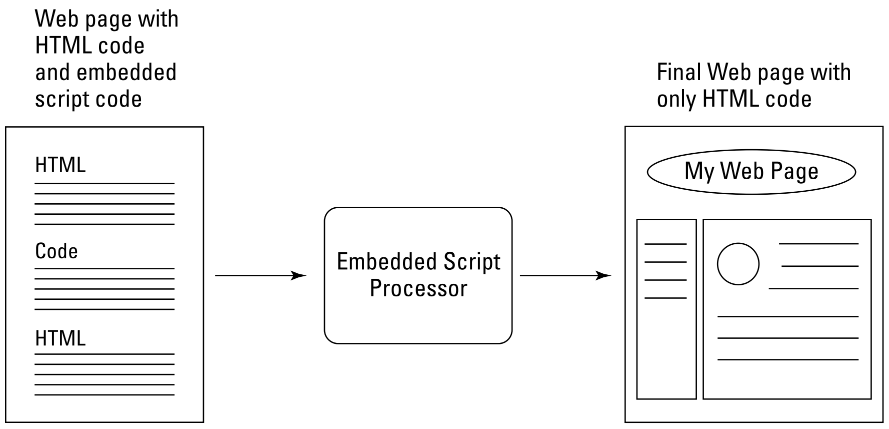

Web page with HTML code and embedded script code
HTML
Code
HTML
Embedded Script Processor
Final Web page with only HTML code
My Web Page

**FIGURE 1-1:** Program code embedded in a web page.

Now, because programming code is embedded in the web page, something somewhere must run the code to produce the dynamic HTML for the new content. As it turns out, there are two places where the embedded program code can run:

- On the client's computer, after the web browser downloads the web page. This is known as *client-side programming.*
- On the web server before the web page is sent. This is known as *server-side programming.*

This section takes a look at how each of these types of programming differ in creating dynamic content for your website.

## Client-side programming

In client-side programming, you embed program code inside the HTML code that the server sends to the client browser with the HTML code. The browser must be able to detect the embedded program code and run it, either inside the browser or as a separate program outside the browser. Figure 1-2 demonstrates this process.

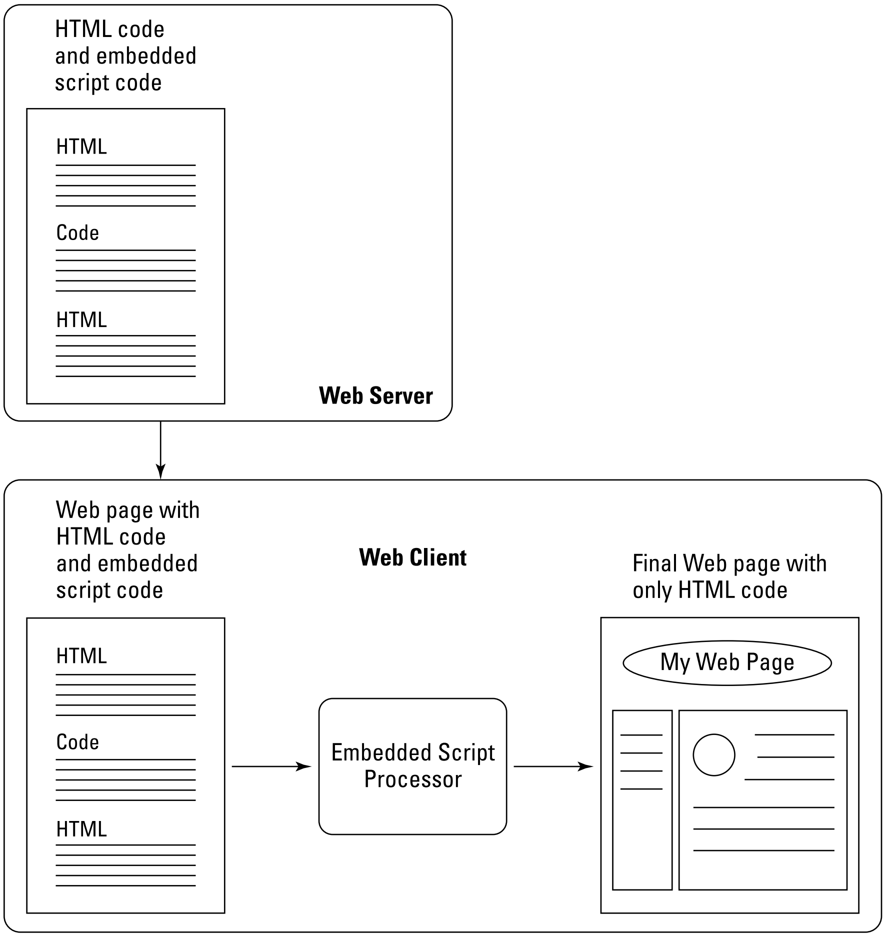

HTML code and embedded script code
HTML
Code
HTML
Web Server
Web page with HTML code and embedded script code
HTML
Code
HTML
Web Client
Embedded Script Processor
Final Web page with only HTML code
My Web Page

**FIGURE 1-2:** Using client-side code in a web page.

### JavaScript

These days, the most popular client-side programming language is JavaScript. JavaScript is a scripting language that you embed inside the normal HTML code in your web page. It runs within the client browser and can utilize features of the browser that are not normally accessible from standard HTML code. JavaScript code is commonly used to produce pop-up messages and dialog boxes that people interact with as they view the page. These are elements that HTML code can't generate.

As shown in Figure 1-2, the entire web page with the embedded JavaScript code is downloaded to the client browser. The client browser detects the embedded JavaScript code and runs it accordingly. It does this while also processing the HTML tags within the document and applying any CSS styles defined. That's a lot for the browser to keep up with!

The downside of JavaScript is that, because it runs in the client browser, you're at the mercy of how the individual web browser interprets the code. Although the HTML language started out as a standard, JavaScript was a little different. In the early days of JavaScript, different browsers would implement different features of JavaScript using different methods. It was not uncommon to run across a web page that worked just fine for one type of browser, but didn't work at all in another type of browser — all because of JavaScript processing inconsistencies.

Eventually, work was done to standardize JavaScript. The JavaScript language was taken up by the Ecma International standards organization, which created the ECMAScript standard, which is what JavaScript is now based off of. As the ECMAScript standard evolved, more and more browser developers started seeing the benefits of using a standard client-side programming language and incorporated them in their JavaScript implementations. At the time of this writing, the eighth version of the standard, called ECMAScript 2017, has been finalized and implemented in most browsers.

TECHNICAL STUFF

The name JavaScript was chosen to capitalize on the popularity of the Java programming language for use in web applications. However, it doesn't have any resemblance or relation to the Java programming language.

## jQuery

JavaScript is popular, but one of its downsides is that it can be somewhat complicated to program. With so many different features incorporated by so many different developers, today a JavaScript program can quickly turn into a large endeavor to code.

To help solve this issue, a group of developers banded together to create a set of libraries to make client-side programming with JavaScript easier. Thus was born jQuery.

The jQuery software isn't a separate programming language; instead, it's a set of libraries of JavaScript code. The libraries are self-contained JavaScript functions that you can reference in your own JavaScript programming to perform common functions, such as finding a location in a web page to display text or retrieve a value entered into an HTML form field.

Instead of having to write lines and lines of JavaScript code, you can just reference one or two jQuery functions to do the work for you. That's a huge time-saver, as well as a great resource for implementing advanced features that you would never have been able to code yourself using just JavaScript.

## Server-side programming

The other side of web programming is server-side programming. Server-side programming languages solve the problem of different client code interpreters by running the code on the server. In server-side programming, the web server interprets the embedded programming code before sending the web page to the client's browser. The server then takes any HTML that the programming code generates and inserts it directly into the web page before sending it out to the client. The server does all the work running the scripting code, so you're guaranteed that every web page will run properly. Figure 1-3 illustrates this process.

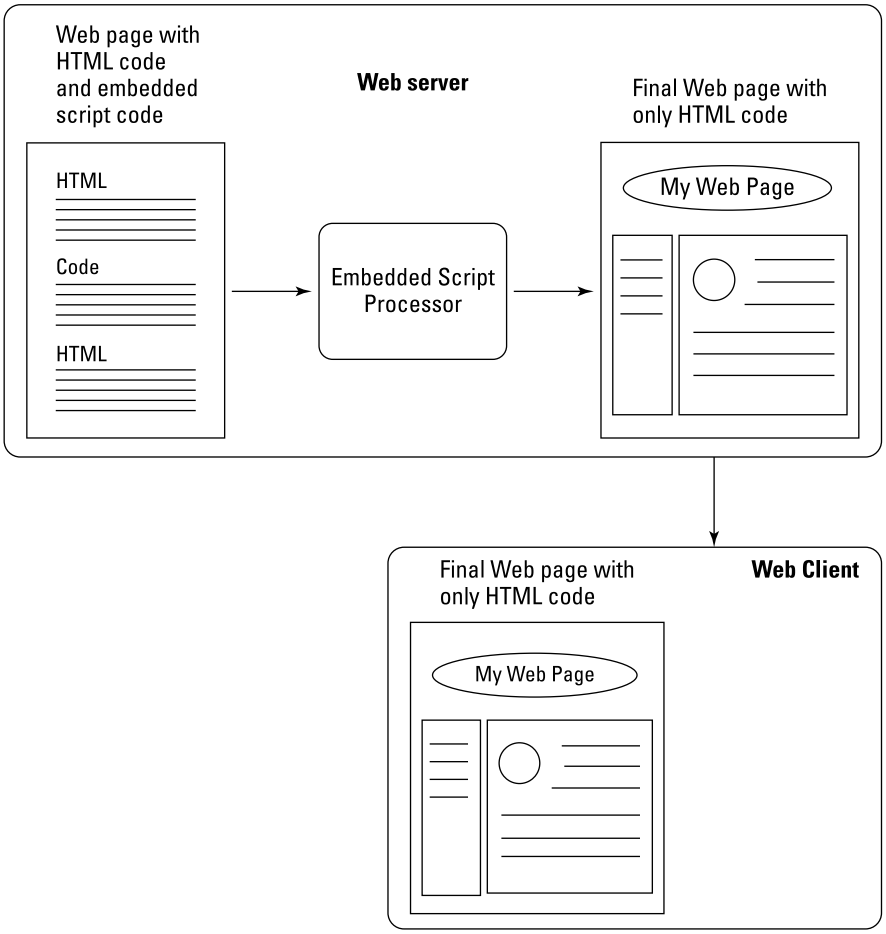

Web page with HTML code and embedded script code
Web server
Final Web page with only HTML code
HTML
Code
HTML
Embedded Script Processor
My Web Page
Final Web page with only HTML code
Web Client
My Web Page

**FIGURE 1-3:** Using server-side programming to create a web page.

Unlike client-side programming, there are many popular server-side programming languages that are in use these days, each with its own set of pros and cons. This section takes a look at a few of the more popular programming languages.

## CGI scripting

One of the first attempts at server-side programming support was the Apache web server's Common Gateway Interface (CGI). The CGI provided an interface between the web server and the underlying server operating system (OS), which was often Unix-based.

This allowed programmers to embed scripting code commonly used in the Unix platform to dynamically generate HTML. Two of the most common scripting languages used in the Unix world and, thus, commonly used in CGI programming are Perl and Python.

Although CGI programming became popular in the early days of the web, it wasn't long before it was exploited. It was all too easy for a novice administrator to apply the wrong permissions to CGI scripts, allowing a resourceful attacker to gain privileged access to the server. Other methods of processing server-side programming code had to be developed.

## Java

One of the earlier attempts at a controlled server-side programming language was Java. Although the Java programming language became popular as a language for creating stand-alone applications that could run on any computer platform, it can also run as a server-side programming language in web applications. When used this way, it's called Java Server Pages (JSP).

The JSP language requires that you have a Java compiler embedded with your web server. The web server detects the Java code in the HTML code and then sends the code to the Java compiler for processing. Any output from the Java program is sent to the client browser as part of the HTML document. The most common JSP platform is the open-source Apache Tomcat server.

## The Microsoft ASP.NET family

Microsoft's first entry into the server-side programming world — Active Server Pages (ASP) — had a similar look and feel to JSP. ASP programs embedded ASP scripting code inside standard HTML code and required an ASP server to be incorporated with the standard Microsoft Internet Information Services (IIS) web server to process the code.

However, Microsoft developers determined that it wasn't necessary to maintain a separate programming language for server-side web programming, so they combined the server-side programming and Windows desktop programming environments into one technology. With the advent of the .NET family of programming languages, Microsoft released ASP.NET for the web environment, as an update to the old ASP environment.

With ASP.NET, you can embed any type of Microsoft .NET programming code inside your HTML documents to produce dynamic content. The .NET family of programming languages includes Visual Basic .NET, C#, J#, and even Delphi. NET. This allows you to leverage the same code you use to create Windows desktop applications as you do to create dynamic web pages. You can often use the same Windows features, such as buttons, slide bars, and scrollbars, inside your web applications that you see in Windows applications.

### JavaScript

Yes, you read that right. The same JavaScript language that's popular in the client-side programming world is now starting to make headway as a server-side programming language. The Node.js library allows you to interface JavaScript code inside HTML web pages for processing on the server.

The benefit to using Node.js is that you only need to learn one language for both client-side and server-side programming. Although it's still relatively new to the game, the Node.js language is becoming more popular.

### PHP

What started out as a simple exercise in tweaking CGI scripts turned into a new server-side programming language that took the world by storm. Rasmus Lerdorf wrote the Personal Home Page (PHP) programming language as a way to improve how his CGI scripts worked. After some encouragement and help, PHP morphed into its own programming language, and a new name, PHP: Hypertext Preprocessor (yes, it uses the acronym inside its name, which is called a *recursive acronym*).

The PHP language developers freely admit that they borrowed many features from other popular languages, such as Perl, Python, C, and even Unix shell scripting. However, PHP was developed specifically for server-side programming, and it has many features built in that aren't available in other scripting languages. You don't need to wrestle with strange setups or features to get PHP to work in a web environment. It has matured into a complete catalog of advanced features that cover everything from database access to drawing graphics on your web page.

Because of the dedication of the PHP developers to create a first-rate server-side programming language, and because it's free open-source software, PHP quickly became the darling of the Internet world. Many web-hosting companies include PHP as part of their basic hosting packages. If you already have space on a web-hosting server, it's possible that you already have access to PHP!

## Combining client-side and server-side programming

Client-side and server-side programming both have pros and cons. Instead of trying to choose one method of creating dynamic web pages, you can instead use both at the same time!

You can easily embed both client-side and server-side programming code into the same web page to run on the server, as shown in Figure 1-4.

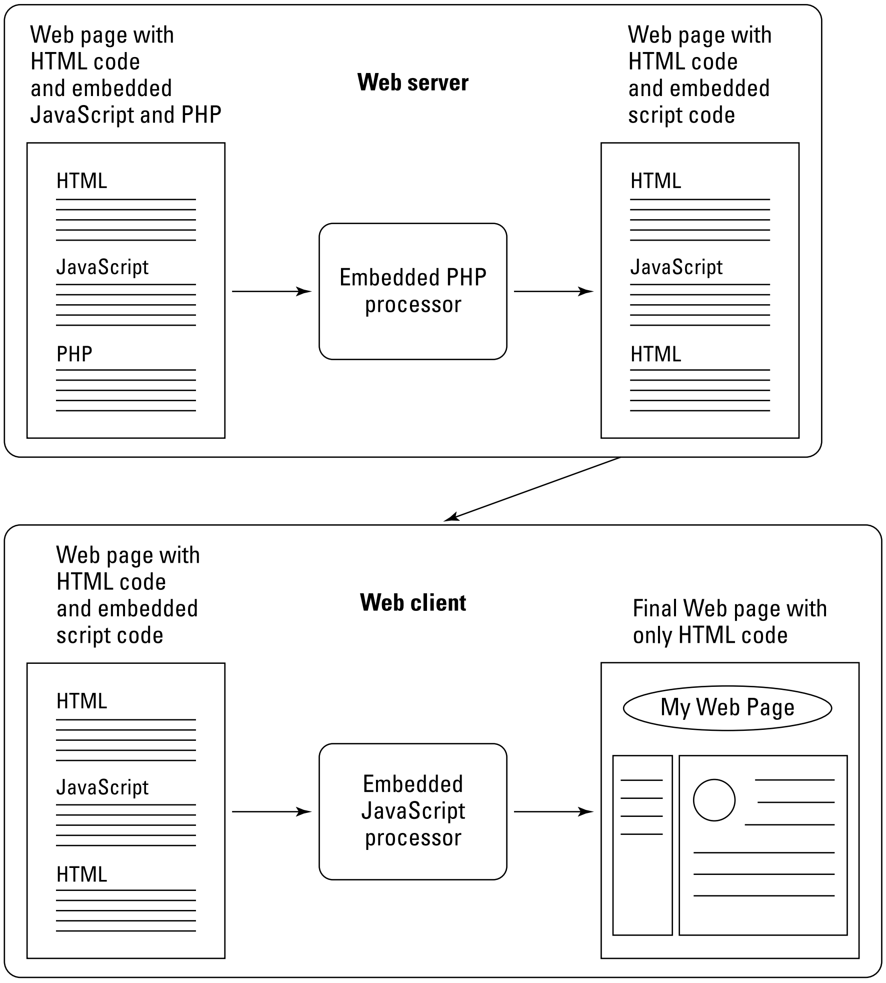

Web server
Web page with HTML code and embedded JavaScript and PHP
HTML
JavaScript
PHP
Embedded PHP processor
Web page with HTML code and embedded script code
HTML
JavaScript
HTML
Web client
Web page with HTML code and embedded script code
HTML
JavaScript
HTML
Embedded JavaScript processor
Final Web page with only HTML code
My Web Page

**FIGURE 1-4:** Combining client-side and server-side programming.

One common use for JavaScript and PHP coding is data validation. When you provide an HTML form for your website visitors to fill out, you have to be careful that they actually fill in the correct type of data for each field. With server-side programming, you can't validate the data until the site visitor completes and submits the form to the server. If a website visitor accidentally skips filling out a single field and the entire form needs to be filled out all over again, that can be a frustrating experience.

To solve that problem, you can embed JavaScript code into the form to check as the site visitor enters data into the form. If any form fields are empty when the Submit button is clicked, the JavaScript code can block the form submission and point out the empty field. Then, when all the data is completed and the form is successfully submitted, the PHP code on the server can process the data to ensure it's the correct data type and format.

## Storing Content

The last piece of the dynamic web application puzzle is the actual content. With static web pages, content is already built into the web page code. To change information on a static web page, you have to recode the page. Unfortunately, more often than not, when a web page is updated, the old version is lost.

With dynamic web applications, the content comes from somewhere outside of the web page. But where? The most common place is a database.

Databases are an easy way to store and retrieve data. They're quicker than storing data using standard files, and they provide a level of security to protect your data. By storing content in a database, you can also easily archive and reference old content and replace it with new content as needed.

Much like the server-side programming world, the database world has lots of different database software options. Here are some of the more popular ones:

- **Oracle:** Oracle has set the gold standard for databases. It's found in many high-profile commercial environments. Although Oracle is very fast and supports lots of features, it can also be somewhat expensive.
- **Microsoft SQL Server:** Microsoft's entry into the database server world, SQL Server is geared toward high-end database environments. It's often found in environments that utilize Microsoft Windows Servers.

- **PostgreSQL:** The PostgreSQL database server is an open-source project that attempts to implement many of the advanced features found in commercial databases. In its early days, PostgreSQL had a reputation for being somewhat slow, but it has made vast improvements. Unfortunately, old reputations are hard to shake, and PostgreSQL still struggles with overcoming them.
- **MySQL:** The MySQL database server is yet another open-source project. Unlike PostgreSQL, it doesn't attempt to match all the features of commercial packages. Instead, it focuses on speed. MySQL has a reputation for being very fast at simple data inserts and queries — perfect for the fast-paced web application world.

  Mainly because of its speed, the MySQL database server has become a popular tool for storing data in dynamic web applications. It also helps that, because it's an open-source project, web-hosting companies can install it for free, which makes it a perfect combination with the PHP server-side programming language for dynamic web applications.

IN THIS CHAPTER

» **Exploring your development options**

» **Picking a development environment**

» **Configuring the servers**

# Chapter 2 Using a Web Server

Before you can start developing dynamic web applications, you'll need a web server environment to work in. You have lots of different choices available to create your own development environment, but sometimes having more options just makes things more confusing. This chapter walks through the different options you have for creating your development environment.

## Recognizing What's Required

Just like that famous furniture that needs assembly, you'll need to assemble some separate components to get your web application development environment up and running. There are three main parts that you need to assemble for your web development environment:

- A web server to process requests from browsers to interact with your application
- A PHP server to run the PHP server-side programming code in your application
- A database server to store the data required for your dynamic application

On the surface, this may seem fairly simple, but to make things more complicated, each of these parts has different options and versions available. That can lead to literally hundreds of different combinations to wade through!

This section helps you maintain your sanity by taking a closer look at each of these three requirements.

## The web server

The web server is what interacts with your website visitors. It passes their requests to your web application and passes your application responses back to them. The web server acts as a file server — it accepts requests for PHP and HTML files from client browsers and then retrieves those files and sends them back to the client browser. As I explain in the preceding chapter, the web server uses the HTTP standard to allow anonymous requests for access to the files on the server and respond to those requests.

There are quite a few different web server options around these days. Here are a few of the more popular ones that you'll run into:

- **Apache:** The granddad of web servers, Apache was derived from the original web server developed at the University of Illinois. It's an open-source software project that has been and is currently the most commonly used web server on the Internet. It is very versatile and supports lots of different features, but with versatility comes complexity. Trying to wade through the configuration file for an Apache web server can be confusing. But for most web environments you just need to change a few of the default configuration settings.
- **nginx:** The newer kid on the block, nginx is intended to ease some of the complexity of the Apache web server and provide improved performance. It's currently gaining in popularity, but it still has a long way to go to catch up with Apache.
- **lighthttpd:** As its name suggests, lighthttpd is a lightweight web server that's significantly less versatile and complex than the Apache web server. It works great for small development environments and is becoming popular in embedded systems that need a web server with a small footprint. However, it doesn't hold up well in large-scale production Web server environments and probably isn't a good choice for a web development environment.
- **IIS:** IIS is the official Microsoft Web server. It's popular in Microsoft Windows server environments, but there aren't versions for other operating systems. IIS focuses on supporting the Microsoft .NET family of server-side programming languages, such as C# .NET and Visual Basic .NET, but it can be interfaced with the PHP server. This configuration is not common, though, and you don't see very many PHP servers that utilize the IIS web server.

As you can tell from these descriptions, just about every web server is compared to the Apache web server. Apache has become the gold standard in Internet web servers. Unless you have a specific reason for not using the Apache web server, you should use it for your development environment, especially if you know that your production web server environment will use it.

## The PHP server

The PHP programming language began in 1995 as a personal project by Rasmus Lerdorf to help his web pages access data stored in a database. He released the first official version 1.0 to the open-source community on June 8, 1995.

Since then, the PHP language has taken on a life of its own, gaining in both features and popularity. The development of the PHP language is currently supported by Zend, which produces many PHP tools.

One of the most confusing aspects of the PHP server is that there are currently two different actively supported branches of the PHP language:

- The version 5.x branch
- The version 7.x branch

The first question that often comes to mind is: "What happened to version 6?" The short-lived version 6 of PHP had some unresolvable issues and was officially abandoned by the PHP developers, with the new features rolled back into version 5.

Now for the second question: "Why two active versions?" The version 5.x branch is still maintained mainly because of the great wealth of applications that continue to use features supported in version 5.x, but not in version 7.x. It will take some time before all the old 5.x applications will be migrated to version 7.x code. Unfortunately, version 7 of PHP breaks quite a few things that were popular in the 5.x version. However, the PHP developers are no longer performing bug fixes in the 5.x branch, only security patches. At the time of this writing, the current version in the 5.x branch is 5.4 and will be maintained until the end of 2018.

At the time of this writing, many popular web server packages support both the 5.x and 7.x version branches and will give you the choice of which one to use for your installation. If you're developing new dynamic web applications, it's best to use the 7.x version branch; at the time of this writing, the latest version is 7.2.

The PHP server contains its own built-in web server, but that's only intended for development and not for use as a live production web server. For large-scale use, you must interface the PHP server with a web server. As the web server receives requests for `.php` files, it must pass them to the PHP server for processing. You must set up this feature as part of the web server configuration file. This is discussed later in this chapter in the "Customizing the Apache Web Server" section.

WARNING

You may still run into some web-hosting companies that use PHP version 4. This was a very popular and long-running version, but it's no longer supported by PHP with security patches. It's best to stay away from any web host that only supports PHP version 4.

## The database server

As I describe in Chapter 1 of this minibook, there are many different types of database servers to handle data for your web applications. By far the most popular used in open-source web applications is the MySQL server.

Many websites and web packages use the term *MySQL Server*, but there are actually a few different versions of it. Because Oracle acquired the MySQL project in 2010, it has split the project into four versions:

- **MySQL Standard Edition:** A commercial product that provides the minimal MySQL database features.
- **MySQL Enterprise Edition:** A commercial product that provides extra support, monitoring, and maintenance features.
- **MySQL Cluster Carrier Grade Edition:** A commercial product that in addition to the Enterprise Edition features, supports multi-server clustering.
- **MySQL Community Edition:** The freely downloadable version of MySQL that supports the same features as the Standard Edition, but with no formal support.

As you can see from the list, the MySQL server has both commercial and open-source versions. The commercial versions support some advanced features that aren't available in the Open Source version, such as hot backups, database activity monitoring, and being able to implement a read/write database cluster on multiple servers. These advanced features can come in handy in large-scale database environments, but for most small to medium-size database applications, the MySQL Community Edition is just fine. That's what's usually included in most web server packages.

Just as with PHP, the MySQL project maintains multiple versions of the MySQL server software. At the time of this writing, the currently supported versions of MySQL are

- Version 5.5
- Version 5.6
- Version 5.7

TECHNICAL STUFF

### MySQL AND MariaDB

The MySQL server project has had quite an interesting life. It was originally developed in 1994 as an open-source project by a Swedish company, MySQL AB. It gained in popularity and features, until MySQL AB was purchased by Sun Microsystems in 2008. However, Oracle purchased Sun Microsystems in 2010 and took control over the MySQL project.

When Oracle purchased the rights to MySQL from Sun Microsystems, the main MySQL developer and his team left to start their own separate open-source branch of MySQL, called MariaDB. With the terms of the open-source license, this move was completely legal, and the project has gained some respect and following in the open-source community. MariaDB is nearly 100 percent compatible with MySQL and is often used as a direct replacement for the MySQL Community Edition in some environments. Any PHP code that you write to interact with the MySQL server will also work with the MariaDB server. Don't be alarmed if the development environment you use switches to MariaDB!

Each version has some minor updates to the MySQL database engine, but for most dynamic web applications, the differences won't play a significant role in your application performance or functions, so it won't matter much which of these three versions your system uses.

Several cloud providers (including Oracle itself) provide the MySQL server as a cloud service. Instead of installing and running your own MySQL server you can rent space on their MySQL cloud server. The benefit of running MySQL in the cloud is that you're guaranteed perfect up-time for the database, because it's distributed among multiple servers in the cloud. The downside, though, is that this can get expensive and is only recommended for commercial web applications that require the extra server power provided by the cloud.

## Considering Your Server Options

Now that you know you'll need a web server, a PHP server, and a MySQL server for your development work, the next step is trying to find an environment that supports all three (and it would help if they were all integrated). You basically have three options for setting up a complete web programming development environment:

- Purchase space on a commercial server from a web-hosting company.
- Install the separate servers on your own workstation or server.
- Install an all-in-one package that bundles all three servers for you.

The following sections walk you through each of these scenarios and the pros and cons of each.

## Using a web-hosting company

By far, the easiest method of setting up a PHP programming environment is to rent space on an existing server that has all the necessary components already installed. Plenty of companies offer PHP web development packages. Some of the more popular ones are

- GoDaddy (`www.godaddy.com`)
- HostGator (`www.hostgator.com`)
- 1&1 (`www.1and1.com`)
- 000webhost (`www.000webhost.com`)

These large web-hosting companies offer multiple levels of support for their services. Often, they'll offer several tiers of service based on the number of databases you can create, the amount of data that you can store, and the amount of network bandwidth your web applications are allowed to consume per month. That way, you can start out with a basic package for minimal cost and then upgrade to one of the more expensive packages as your Internet application takes off! It pays to shop around to check different pricing structures and support levels at the different web-hosting companies.

Besides these main competitors, you'll find many, many smaller web hosting companies willing to offer MySQL/PHP packages to host your applications. There's a good chance if you already have a web-hosting company you use to host your static web pages, it'll have some type of MySQL/PHP support. If you already have space on a web server for your website, check with them to see if they offer an upgrade to a MySQL/PHP package.

With the popularity of the new "cloud" environment where everything runs on shared server space, there are now a few more participants in the PHP server hosting game. The Wikipedia web page for cloud service providers lists more than 200 different providers! You'll probably recognize the more popular ones:

- Amazon Web Services (AWS)
- Google Cloud Platform
- Oracle Cloud Platform
- Microsoft Azure

Each of these cloud services provides some level of support for PHP program development. One of the main benefits of utilizing a cloud service is that your application is hosted on multiple servers that share the traffic load and are redundant for backup purposes. If a server in the cloud crashes, your application will still work on other servers. Of course, be prepared to pay a premium price for those features!

WARNING

Be careful with some of the smaller web-hosting companies. These days, just about anyone can install the PHP and MySQL server software onto a server and sell space, so many "mom-and-pop" web-hosting companies now provide this service. However, installing the server programs is different from maintaining the server programs. Often, you'll find these smaller web-hosting sites use outdated server versions that haven't been upgraded or patched with security updates, making them vulnerable to attacks.

## Building your own server environment

I wouldn't recommend it for a live production website, but for development work you can build your own web server environment. You don't even need to have a large server for a personal web development environment — you can build it using your existing Windows or Apple workstation or laptop.

The following sections walk you through the basics you need to know to get this working in either the Linux or Windows/Mac environments.

### Web servers in Linux

Linux desktops and servers are becoming more popular these days, especially for web development. You can download the Apache, MySQL, and PHP server source code packages and compile them on your Linux system, but unless you need the absolute latest version of things, that's not the recommended way to do it.

These days, most Linux distributions include packages for easily installing all the components you need for a complete web development environment. For Debian-based Linux distributions (such as Ubuntu and Linux Mint), you use the `apt-get` command-line tool to install software packages. For Red Hat–based Linux distributions (such as Red Hat, CentOS, and Fedora) you use the `dnf` command-line tool.

For Debian-based systems, such as Ubuntu, follow these steps to do that:

1. **From a command prompt, install the Apache web server using the following command:**

   ```
   sudo apt-get install apache2
   ```

2. **Install the MySQL server package using the following command:**

   ```
   sudo apt-get install mysql-server
   ```

   WARNING

   During the MySQL server installation, you'll be prompted for a password to use for the root user account. The root user account in MySQL has full privileges to all tables and objects. Make sure you remember what password you enter here!

3. **Install the PHP packages to install the PHP server and required extensions, the Apache modifications to run PHP, and the graphical phpMyAdmin tool:**

   ```
   sudo apt-get install php libapache2-mod-php
   sudo apt-get install php-mcrypt php-mysql
   sudo apt-get install phpmyadmin
   ```

   The first line installs the main PHP server, along with the Apache module to interface with the PHP server. The second line installs two PHP extensions that are required to interface with the MySQL server. The third line installs the web-based phpMyAdmin PHP program, which provides a web interface to the MySQL server.

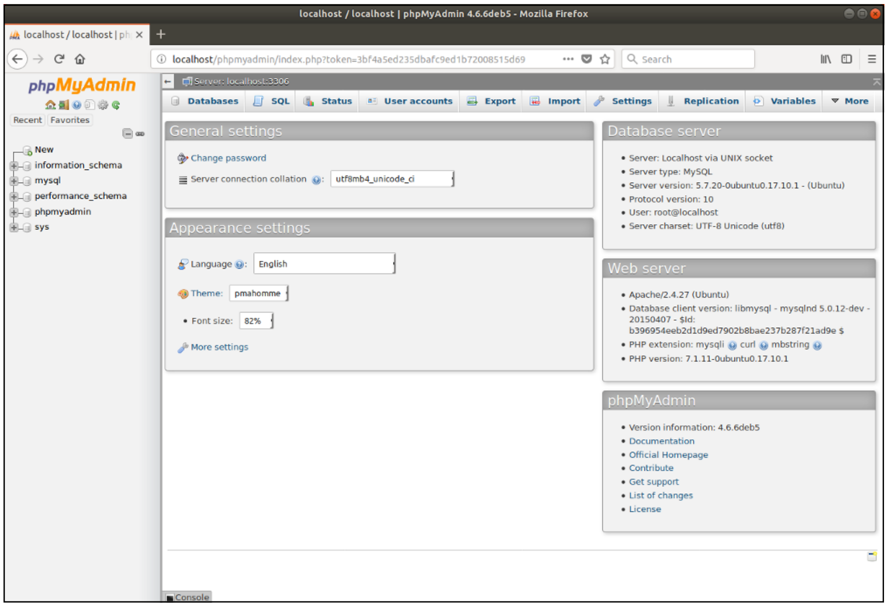
4. **Open a browser and test things out by going to the following URL:**

   ```
   http://localhost/phpmyadmin
   ```

   You should be greeted by the phpMyAdmin administration window.

5. **Log in using the MySQL root user account and the password you supplied when you installed MySQL (you remember it, right?).**

   Figure 2-1 shows the main phpMyAdmin web page, which shows what versions of the Apache, PHP, and MySQL servers are running.



General settings

- Change password
- Server connection collation: utf8mb4_unicode_ci

Appearance settings

- Language: English
- Theme: pmahomme
- Font size: 82%
- More settings

Database server

- Server: Localhost via UNIX socket
- Server type: MySQL
- Server version: 5.7.20-0ubuntu0.17.10.1 - (Ubuntu)
- Protocol version: 10
- User: root@localhost
- Server charset: UTF-8 Unicode (utf8)

Web server

- Apache/2.4.27 (Ubuntu)
- Database client version: libmysql - mysqlnd 5.0.12-dev - 20150407 - $Id: b396954eeb2d1d9ed7902b8bae237b287f21ad9e $
- PHP extension: mysqli curl mbstring
- PHP version: 7.1.11-0ubuntu0.17.10.1

phpMyAdmin

- Version information: 4.6.6deb5
- Documentation
- Official Homepage
- Contribute
- Get support
- List of changes
- License

**FIGURE 2-1:** The main phpMyAdmin web page showing everything that is running.

For Red Hat–based systems, such as Fedora and CentOS, follow these steps to load LAMP:

1. **From a command prompt, install the Apache web server using the following commands:**

```
sudo dnf install httpd
sudo systemctl enable httpd
sudo systemsctl start httpd
```

   The `httpd` package includes the Apache2 web server. The executable file for Apache is named `httpd` (thus, the name of the package). The package doesn't start the Apache web server by default, so the second two lines use the `systemctl` utility to enable the service so it starts automatically at boot time and then starts it.

2. **Install the MySQL server package using the following commands:**

```
sudo dnf install mariadb-server
sudo systemctl enable mariadb
sudo systemctl start mariadb
```

Notice that the Red Hat distribution (and thus CentOS and Fedora) has gone with the MariaDB replacement package for MySQL. When you install MariaDB, the package sets the root user account password to an empty string. This is not recommended if your server is on any type of a network. Fortunately, there's a quick utility that you can run to change the root user account's password:

```
mysql_secure_installation
```

When you run this script, it'll prompt you to answer a few questions, such as the new password for the root user account, whether to restrict the root user account to only logging in from the local host, whether to remove the anonymous users feature, and whether to remove the test database.

3. **Install the PHP packages using the following commands:**

```
sudo dnf install php php-mbstring php-mysql
sudo dnf install phpmyadmin
sudo systemctl restart httpd
```

The PHP server doesn't run as its own service — the Apache web server spawns it when needed. Because of that, you do need to use the `systemctl` utility to restart the Apache web server so it rereads the configuration file with the new PHP settings.

4. **Open a browser and test things out by going to the following URL:**

```
http://localhost/phpmyadmin
```

You should see the phpMyAdmin login page.

5. **Log in using the `root` user account in MySQL along with the password you defined when you installed MySQL.**

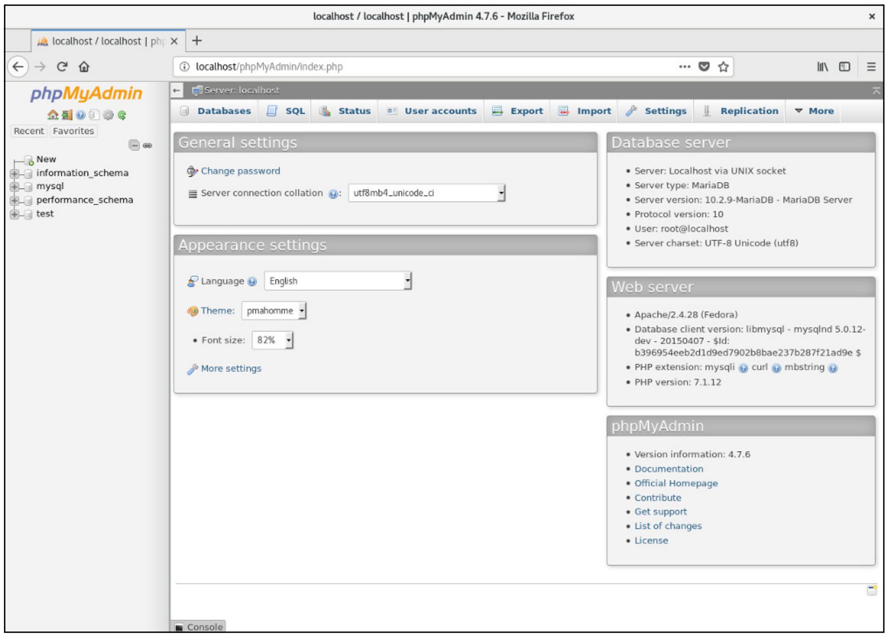
Figure 2-2 shows phpMyAdmin running on a CentOS 7 system.

TIP

By using the distribution software packages for each server, you're guaranteed that the server will run correctly in your Linux environment. An additional benefit is that the distribution software updates will include any security patches or bug fixes released for the servers automatically.

## Web servers in Windows and Mac

Installing and running the Apache, MySQL, and PHP servers in a Windows or Mac environment is very tricky, because there are lots of factors involved in how to install and configure them. For starters, both Windows and macOS come with a web server built in, so if you install the Apache web server you'll need to configure it to use an alternative TCP port.



General settings

- Change password
- Server connection collation: utf8mb4_unicode_ci

Appearance settings

- Language: English
- Theme: pmahomme
- Font size: 82%
- More settings

Database server

- Server: Localhost via UNIX socket
- Server type: MariaDB
- Server version: 10.2.9-MariaDB - MariaDB Server
- Protocol version: 10
- User: root@localhost
- Server charset: UTF-8 Unicode (utf8)

Web server

- Apache/2.4.28 (Fedora)
- Database client version: libmysql - mysqlnd 5.0.12-dev - 20150407 - $Id: b396954eeb2d1d9ed7902b8bae237b287f21ad9e $
- PHP extension: mysqli curl mbstring
- PHP version: 7.1.12

phpMyAdmin

- Version information: 4.7.6
- Documentation
- Official Homepage
- Contribute
- Get support
- List of changes
- License

**FIGURE 2-2:** Viewing the phpMyAdmin main web page on Fedora 27.

Likewise, macOS includes an older version of PHP by default, so if you install an updated version of PHP, things get tricky trying to make sure which version is active.

Because of these complexities, it's not recommended for beginners to install the Apache, MySQL, and PHP packages separately in the Windows and Mac environments. There's a much simpler way of getting that to work, which I'll describe in the next section.

## Using premade servers

Trying to get a working Apache, MySQL, and PHP server in Windows (called WAMP) or in the Mac environment (called MAMP) can be a complicated process. There's a lot of work downloading each of the individual server packages, configuring them, and getting things to work together.

Fortunately, some resourceful programmers have done that work for us! There are quite a few open-source packages that bundle the Apache web server, MySQL (or MariaDB) server, and PHP server together to install as a single package. This is by far the best way to go if you plan on using your Windows or Mac workstation or laptop as your web development environment.

There are quite a few pre-loaded packages available, but these are the most common ones:

- **XAMPP:** An all-in-one package that supports PHP and Perl server-side programming and also includes an email and FTP server, along with a self-signed certificate to use the Apache web server in HTTPS mode. It has installation packages available for Windows, Mac, and Linux.
- **Wampserver:** A Windows-based all-in-one package that allows you to install multiple versions of the Apache, MySQL, and PHP servers at the same time. You can then mix-and-match which versions of which server you have active at any time, allowing you to duplicate almost any web-hosting environment.
- **MAMP:** A Mac-based all-in-one package that is easy to install and use. It also has a commercial package called MAMP Pro that provides additional features for managing your web environment for professional developers.

Of these, the XAMPP package is by far the most popular. It was created by the Apache Friends organization to help promote the use of the Apache web server in web development environments. Follow these steps to install XAMPP in a Windows or macOS environment:

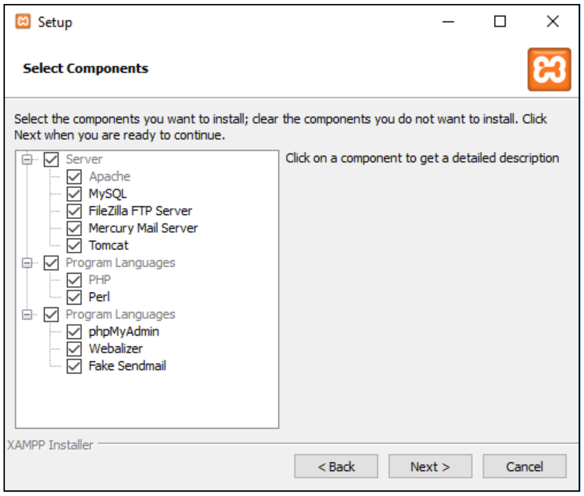
1. **Open your browser and go to `www.apachefriends.org`.**
2. **Look for the Download section of the web page and click the link for the OS you're using.**
3. **After the download finishes, run the downloaded file in your OS environment.**

   This starts the XAMPP installation wizard.
4. **Click the Next button to go to the Select Components window, shown in Figure 2-3.**

   The Select Components window allows you to select which components in XAMPP you want installed. You won't use everything contained in XAMPP for this book, but feel free to install the entire package and explore on your own!
5. **Click the Next button to continue the installation.**
6. **Select the installation folder for XAMPP.**

   The default location for Windows is `c:\xampp`; for macOS, it's `/Applications/XAMPP`. Those will work just fine for both environments.
7. **Click the Next button to continue the installation.**

   The Apache Friends organization has teamed up with Bitnami, which has prepackaged many popular web applications specifically for use in XAMPP.



Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

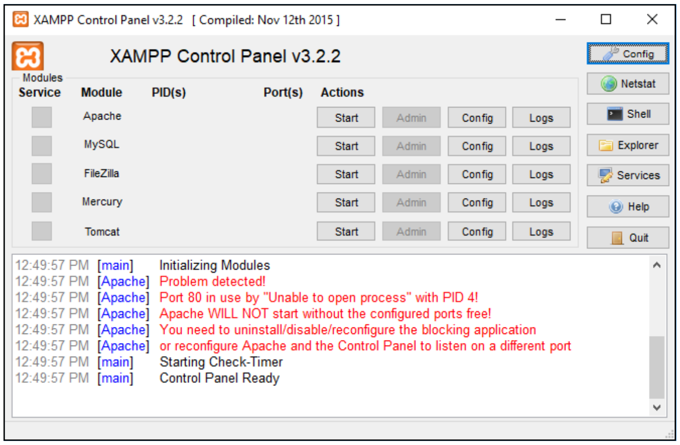
- ☑ Server
  - ☑ Apache
  - ☑ MySQL
  - ☑ FileZilla FTP Server
  - ☑ Mercury Mail Server
  - ☑ Tomcat
- ☑ Program Languages
  - ☑ PHP
  - ☑ Perl
- ☑ Program Languages
  - ☑ phpMyAdmin
  - ☑ Webalizer
  - ☑ Fake Sendmail

Click on a component to get a detailed description

**FIGURE 2-3:** The XAMPP Select Components window in the installation wizard.

8. **You can learn more about Bitnami by leaving the Learn More about Bitnami for XAMPP check box checked, or if you'd like to skip this step, remove the check mark from the check box, and then click the Next button to continue.**
9. **Click the Next button to begin the software installation.**
10. **You can keep the check mark in the check box to start XAMPP, and then click the Finish button to end the wizard.**

The XAMPP Control Panel provides easy access to start, stop, and configure each of the servers contained in the XAMPP package. Figure 2-4 shows the main Control Panel window.


XAMPP Control Panel v3.2.2 [ Compiled: Nov 12th 2015 ]
XAMPP Control Panel v3.2.2
Modules
Config
Netstat
Shell
Explorer
Services
Help
Quit

| Service | Module | PID(s) | Port(s) | Actions | | | |
|---|---|---|---|---|---|---|---|
| | Apache | | | Start | Admin | Config | Logs |
| | MySQL | | | Start | Admin | Config | Logs |
| | FileZilla | | | Start | Admin | Config | Logs |
| | Mercury | | | Start | Admin | Config | Logs |
| | Tomcat | | | Start | Admin | Config | Logs |

```
12:49:57 PM  [main]     Initializing Modules
12:49:57 PM  [Apache]   Problem detected!
12:49:57 PM  [Apache]   Port 80 in use by "Unable to open process" with PID 4!
12:49:57 PM  [Apache]   Apache WILL NOT start without the configured ports free!
12:49:57 PM  [Apache]   You need to uninstall/disable/reconfigure the blocking application
12:49:57 PM  [Apache]   or reconfigure Apache and the Control Panel to listen on a different port
12:49:57 PM  [main]     Starting Check-Timer
12:49:57 PM  [main]     Control Panel Ready
```

**FIGURE 2-4:** The main XAMPP Control Panel window.

By default, XAMPP configures the Apache web server to use TCP port 80 for HTTP connections. Unfortunately, this port is often in use by web servers built into Windows and Mac workstations and servers. This will produce an error message when you first start the XAMPP Control Panel, as shown in Figure 2-4.

You can move the Apache web server to an alternative TCP port. Just follow these steps:

1. **From the XAMPP Control Panel main window, click the Config button for the Apache web server.**
2. **Select the menu option to edit the `httpd.conf` configuration file.**

   This opens the Apache web server configuration file in a text editor.
3. **Look for the line:**

   ```
   Listen 80
   ```
4. **Change the 80 in the line to 8080, a common alternative TCP port to use for HTTP communications.**
5. **Save the updated configuration file in the editor, and then exit the editor window.**
6. **Click the Start button for the Apache web server.**

   The Apache Web server should indicate that it has started and is using both TCP Ports 443 (for HTTPS) and 8080 (for HTTP).
7. **Click the Start button for the MySQL database server.**

   The MariaDB database server should indicate that it has started and is using TCP Port 3306 (the default TCP port for MySQL).

After the Apache and MySQL servers start, you can exit the XAMPP Control Panel. If you need to stop the servers, reopen the XAMPP Control Panel and click the Stop buttons for both servers.

TIP

Although you've moved the Apache web server in the configuration file, XAMPP will still check to see if TCP Port 80 is available when you start the XAMPP Control Panel and complain that it's not available. To stop that, click the Config button in the Control Panel and then remove the check mark for the Check Default Ports on Startup check box.

## Tweaking the Servers

When you get the Apache, MySQL, and PHP servers installed in your development environment, you may need to do a little bit of tweaking to get them working just the way you want. Each of the servers uses a text configuration file to define just how the server behaves. The following sections walk you through how to find the configuration files and some of the settings that you may need to tweak for your development environment.

### Customizing the Apache Server

By default, the Apache Web server uses the `httpd.conf` configuration file to store its settings. For Linux and Mac systems, the file is usually stored in the `/etc` folder structure, often under either `/etc/httpd` or `/etc/apache2`.

TIP

The XAMPP package installs the Apache configuration file in the `c:\xampp\apache\conf` folder in Windows or `/Applications/XAMPP/apache/conf` in macOS.

The `httpd.conf` configuration file contains individual lines called *directives.* Each directive defines one configuration option, along with the value that you set.

The Apache web server is very versatile, with lots of different options and features. The downside to that is it can make the configuration seem complex at first, but the configuration file is organized such that you should be able to find what you're looking for relatively easily. In the following sections, I cover a few things that you'll want to pay attention to.

WARNING

Many systems break the Apache web server configurations into multiple files to help make the features more modular. Look for the `Include` directive lines in the main `httpd.conf` configuration file to see what other files contain Apache web server configuration settings.

#### Defining the web folder location

The main job of the Apache web server is to serve files to remote clients. However, you don't want just anyone retrieving just any file on your system! To limit what files the Apache server serves, you must restrict it to a specific folder area in the system.

You set the folder where the Apache web server serves files using the `DocumentRoot` directive:

```
DocumentRoot c:/xampp/htdocs
```

The `htdocs` folder is the normal default used for the Apache web server in Windows and macOS environments (for macOS, it's located in `/Applciations/XAMPP/htdocs`). For Linux environments, it has become somewhat common to use `/var/www/html` as the `DocumentRoot` folder.

WARNING

If you choose to move the `DocumentRoot` folder to another folder location on the server, make sure the user account that runs the Apache web server has access to at least read files from the folder.

### Setting the default TCP port

The Apache web server listens for incoming connections from client browsers using two different default TCP network ports:

- TCP port 80 for HTTP requests
- TCP port 443 for HTTPS requests

HTTPS requests use encryption to secure the communication between the browser and the server. This method is quickly becoming a standard for all web servers on the Internet.

You set the ports the Apache web server accepts incoming requests on using the `Listen` directive:

```
Listen 80
Listen 443
```

You can use multiple `Listen` directives in the configuration file to listen on more than one TCP port.

## USING ENCRYPTION

To establish a secure HTTPS connection, your Apache web server must have a valid encryption certificate signed by a *certificate authority.* The certificate authority recognizes your website as valid and vouches for your authenticity. This enables your website visitors to trust that you are who you say you are and that your web server is what it says it is.

Unfortunately, signed certificates must be purchased and can be somewhat expensive. For development work, you can use a *self-signed certificate.* The self-signed certificate is what it says: You sign your own certificate. This doesn't instill any trust in your website visitors, so don't use a self-signed certificate on a production website — only use it for development. The XAMPP web server installs a self-signed certificate just for this purpose!

## Interacting with the PHP server

The Apache web server must know how to pass files that contain PHP code to the PHP server for processing. This is a two-step process.

First, you have to tell the Apache web server to load the PHP server module so that it can establish the link between the Apache and PHP servers. You do that using the `LoadModule` directive:

```
LoadModule php7_module "c:/xampp/php/apache2_4.dll"
```

After Apache loads the PHP module, you have to tell it what type of files to send to the PHP server. You do this using the `AddHandler` directive:

```
AddHandler application/x-httpd-php .php
```

This directive tells the Apache web server to forward all files with the `.php` file extension to the PHP module, which then forwards the files to the PHP server for processing.

WARNING

It may be tempting to just forward all `.html` files to the PHP server, because the PHP server will pass any HTML code directly to the client browser. However, this will add extra processing time to load your static web pages, causing a performance issue with your HTML pages.

## Tracking errors

When working in a development environment, it's always helpful to be able to track any errors that occur in your applications. The Apache web server supports eight different levels of error messages, shown in Table 2-1.

**TABLE 2-1** Apache Web Server Error Levels

| Error Level | Description |
|---|---|
| `emerg` | A fatal error will halt the Apache web server. |
| `alert` | A severe error will have an adverse impact on your application and should be resolved immediately. |
| `crit` | A critical condition caused the operation to fail, such as a failure to access the network. |
| `error` | An error occurred in the session, such as an invalid HTTP header. |
| `warn` | A minor issue occurred in the session but didn't prevent it from continuing. |
| `notice` | Something out of the normal occurred. |
| `debug` | A low-level detailed message occurs for each step the server takes in processing a request. |

You define the level of error tracking using the `LogLevel` directive and the location of the error log using the `ErrorLog` directive:

```
LogLevel warn
ErrorLog logs/error.log
```

The `debug` log level can be useful for troubleshooting issues but is not recommended for normal activity, because it generates lots of output!

TIP

You can customize the appearance of the log messages using the `LogFormat` directive. Apache allows you to determine just what information appears in the log file, which can be handy when trying to troubleshoot specific problems. Consult the Apache server documentation for the different options you have available for customizing the logs.

## Customizing the MySQL server

The MySQL server uses two different filenames for its configuration settings:

- `my.cnf` for Linux and Mac systems
- `my.ini` for Windows systems

One of the more confusing features about the MySQL server is that there are three ways to specify configuration settings:

- They can be compiled into the executable server program when built from source code.
- They can be specified as command-line options when the server starts.
- They can be set in the MySQL configuration file.

You can compile all the settings you need into the MySQL executable server program and run with no configuration file at all (that's the approach the MAMP all-in-one package takes). The downside to that is it's hard to determine just which settings are set to which values.

Most MySQL server installations use a combination of compiling some basic settings into the executable server program and creating a basic configuration file for the rest. The setting values set in the configuration file override anything compiled into the executable server program or set on the command line.

As with the Apache web server, the MySQL database server has lots of options you can change in the configuration file to fine-tune how things work. That said, there are only a few items that you'd ever really need to tweak in a normal setup. The following sections walk you through some of the settings you should become familiar with.

## The core server settings

The core server settings define the basics of how the MySQL server operates. These settings in the XAMPP for Windows setup look like this:

```
[mysqld]
port = 3306
socket = "C:/xampp/mysql/mysql.sock"
basedir = "C:/xampp/mysql"
tmpdir = "C:/xampp/mysql/tmp"
datadir = "C:/xampp/mysql/data"
log_error = "mysql_error.log"
```

The `port` setting defines the TCP port the MySQL server listens for incoming requests on. The `socket` setting defines the location of a socket file that local clients can use to communicate with the MySQL server without using the network.

The `basedir`, `tmpdir`, and `datadir` settings define the locations on the server that MySQL will use for storing its working files. The `datadir` setting defines where MySQL stores the actual database files.

## Working with the InnoDB storage engine

The InnoDB storage engine provides advanced database features for the MySQL server. It has its own set of configuration settings that control exactly how it operates and how it handles the data contained in tables that use that storage engine.

There are two main configuration settings that you may need to tweak for your specific MySQL server installation:

```
innodb_data_home_dir = "C:/xampp/mysql/data"
innodb_data_file_path = ibdata1:10M:autoextend
```

The `innodb_data_home_dir` setting defines the location where MySQL places files required to support the InnoDB storage engine. This allows you to separate those files from the normal MySQL database files if needed.

The `innodb_data_file_path` setting defines three pieces of information for the storage engine:

- The filename MySQL uses for the main InnoDB storage file
- The initial size of the storage file
- What happens when the storage file fills up

To help speed up the data storage process, the InnoDB storage engine pre-allocates space on the system hard drive for the database storage file. That way, for each data record that's inserted into a table, the storage engine doesn't need to ask the operating system for more disk space to add to the database file — it's already there! This greatly speeds up the database performance. The second parameter defines the initial amount of disk space that the InnoDB storage engine allocates.

The third parameter is where things get interesting. It defines what the InnoDB storage engine does when the space allocated for the storage file becomes full. By default, the InnoDB storage engine will block new data inserts to the tables when it runs out of allocated storage space. You would have to manually extend the storage file size.

When you specify the `autoextend` setting, that allows the InnoDB storage engine to automatically allocate more space for the file. That's convenient, but it can also be dangerous in some environments. The InnoDB storage engine will keep allocating more storage space as needed until the server runs out of disk space!

WARNING

When you use the InnoDB storage engine for your MySQL applications, it's always a good idea to keep an eye on the storage space folder to make sure it's not taking up all the server disk space.

## Customizing the PHP server

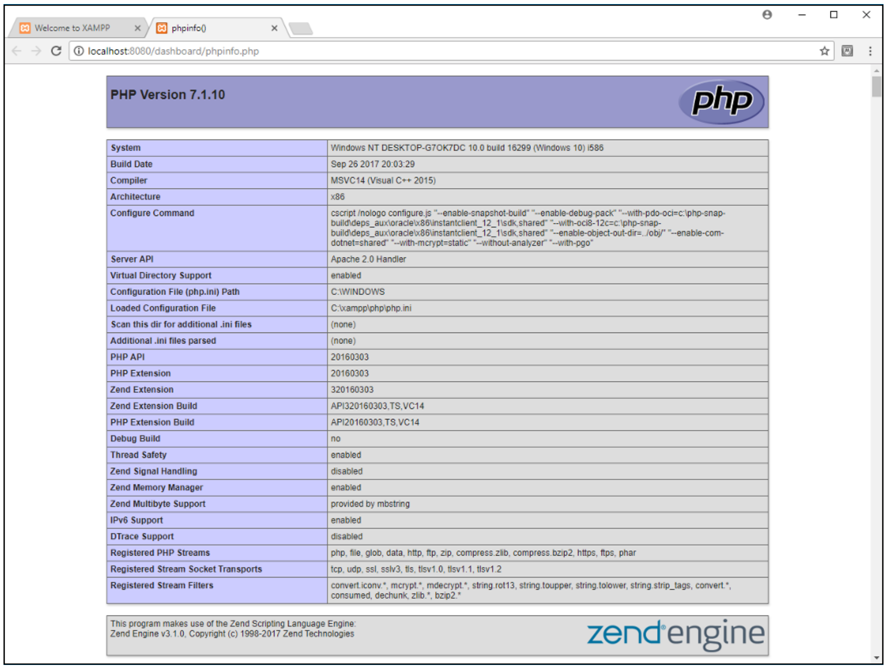
The PHP server configuration file is named `php.ini`, but it can be located in several different areas. The locations that the PHP server checks are (in order):

- The path set in the `PHPIniDir` directive in the Apache web server configuration file
- The path set in a system environment variable named `PHPRC`
- For Windows systems, the path set in the registry key named `IniFilePath` under the `HKEY_LOCAL_MACHINE/Software/PHP` registry hive
- The folder where the PHP server executable file is stored

- The default web server's folder
- The OS system folder, which for Windows is the `c:\winnt` folder, and for Linux and Mac the `/usr/local/lib` folder

The XAMPP install process places the `php.ini` file in the `c:\xampp\apache\bin` folder.

If you're ever in doubt as to which `php.ini` configuration file the PHP server is using, run the `phpinfo()` function in a small PHP program. For your convenience, all the popular all-in-one packages provide a link to run the `phpinfo()` function from their main web pages. Figure 2-5 shows the output from the `phpinfo()` function in XAMPP running on a Windows system.



PHP Version 7.1.10

| | |
|---|---|
| System | Windows NT DESKTOP-G7OK7DC 10.0 build 16299 (Windows 10) i586 |
| Build Date | Sep 26 2017 20:03:29 |
| Compiler | MSVC14 (Visual C++ 2015) |
| Architecture | x86 |
| Configure Command | cscript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-pdo-oci=c:\php-snap-build\deps_aux\oracle\x86\instantclient_12_1\sdk,shared" "--with-oci8-12c=c:\php-snap-build\deps_aux\oracle\x86\instantclient_12_1\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--without-analyzer" "--with-pgo" |
| Server API | Apache 2.0 Handler |
| Virtual Directory Support | enabled |
| Configuration File (php.ini) Path | C:\WINDOWS |
| Loaded Configuration File | C:\xampp\php\php.ini |
| Scan this dir for additional .ini files | (none) |
| Additional .ini files parsed | (none) |
| PHP API | 20160303 |
| PHP Extension | 20160303 |
| Zend Extension | 320160303 |
| Zend Extension Build | API320160303,TS,VC14 |
| PHP Extension Build | API20160303,TS,VC14 |
| Debug Build | no |
| Thread Safety | enabled |
| Zend Signal Handling | disabled |
| Zend Memory Manager | enabled |
| Zend Multibyte Support | provided by mbstring |
| IPv6 Support | enabled |
| DTrace Support | disabled |
| Registered PHP Streams | php, file, glob, data, http, ftp, zip, compress.zlib, compress.bzip2, https, ftps, phar |
| Registered Stream Socket Transports | tcp, udp, ssl, sslv3, tls, tlsv1.0, tlsv1.1, tlsv1.2 |
| Registered Stream Filters | convert.iconv.*, mcrypt.*, mdecrypt.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, zlib.*, bzip2.* |

This program makes use of the Zend Scripting Language Engine:
Zend Engine v3.1.0, Copyright (c) 1998-2017 Zend Technologies

zend engine

**FIGURE 2-5:** The `phpinfo()` function output.

The `phpinfo()` function displays the system values for each of the configuration file settings and if any were overridden by a local setting. Look for the Loaded Configuration File entry that shows the path to the active `php.ini` file to see where that file is located for your PHP server.

As you can imagine, there are lots of settings available in the php.ini configuration file. Here are some of the php.ini settings (and the default values set in XAMPP) that you may need to tweak on your PHP server:

- date.timezone = Europe/Berlin: Defines the time zone of the PHP server. This must use a time zone value defined at http://php.net/manual/en/timezones.php.
- display_errors = On: Defines whether PHP error messages appear on the web page. This feature is extremely handy for development work but should be disabled for production servers.
- error_reporting = E_ALL & ~E_DEPRECATED: Sets the level of error reporting from the PHP server. PHP uses a complicated bit pattern to set which errors to display or not display. It uses labels to indicate the error level and Boolean bitwise operators to combine the levels — the tilde (~) indicates the NOT operator. The error levels are:
  - E_ERROR: Fatal run-time errors
  - E_WARNING: Run-time warnings that won't halt the script
  - E_PARSE: Parsing syntax errors
  - E_NOTICE: Script encountered something that could be an error and effect the results
  - E_CORE_ERROR: Fatal error that prevents PHP from starting
  - E_CORE_WARNING: Non-fatal errors during startup
  - E_COMPILE_ERROR: Fatal error while compiling the PHP code
  - E_COMPILE_WARNING: Non-fatal errors during compile time
  - E_USER_ERROR: Fatal error message generated manually by your PHP code
  - E_USER_WARNING: Non-fatal error message generated manually by your PHP code
  - E_USER_NOTICE: Notice message generated manually by your PHP code
  - E_STRICT: PHP detected code that doesn't follow the PHP strict rules
  - E_RECOVERABLE_ERROR: A fatal error that you can catch with a try-catch block
  - E_DEPRECATED: The PHP parser detected code that will no longer be supported
  - E_USER_DEPRECATED: A deprecation error generated manually by your PHP code
  - E_ALL: All errors and warnings except E_STRICT

- `variables_order = "GPCS"`: The order in which PHP populates the data from the HTTP session (G = `GET`, P = `POST`, C = Cookies, and S = System variables)
- `short_open_tag = Off`: Determines if you can use the `<?` tag to identify PHP code in your HTML documents
- `max_execution_time = 30`: Sets a time limit (in seconds) for a PHP program to run before the PHP server kills it (This is useful for stopping programs stuck in a loop!)
- `memory_limit = 128M`: Sets a limit on how much memory on the physical server the PHP server can allocate (This also helps prevent runaway programs from taking down the entire web server!)

IN THIS CHAPTER

» Identifying which tools to avoid

» Finding the right tools for the job

# Chapter 3 Building a Development Environment

When you're ready to start coding your web application, you'll need some tools to help you out. Just as a carpenter needs a set of tools to do her job, web developers need tools as well. And just as the carpenter has a wide selection of tools to choose from, so do web developers. A carpenter can build an entire house using a hammer and hand saw (and possibly a tape measure), but most likely, she has a few more advanced tools to make her job easier. Likewise, you can build an entire web application using a standard text editor, but there are plenty of other tools around to make your job easier. The trick to becoming comfortable with web programming is to find the right tool, or combination of tools, for the task at hand. This chapter walks you through some of the tools that you can use to help make your programming job easier. But first, I start by telling you what *not* to use.

## Knowing Which Tools to Avoid

Before I get too far into the tool discussion, I need to tell you what tools *not* to use for serious web-programming jobs. These days, plenty of tools are available to help novice web designers create their own web pages without doing any coding at all. However, trying to develop dynamic web applications with these tools can create more problems than they're worth. Here are some of the tools you should avoid.

## Graphical desktop tools

Graphical desktop tools allow you to create a web page using a purely graphical interface, without having to do any coding. The most popular of these tools are the Microsoft Expression Web package and Adobe Dreamweaver.

Both of these tools use the *what you see is what you get* (WYSIWYG) method of creating web pages. Instead of an editor for writing code, the tool presents you with a graphical canvas that represents your web page. To add features to the web page, you drag and drop elements like text, menus, images, or multimedia clips onto the canvas. When you've created the web page layout, you click a button and the tool automatically generates all the HTML and CSS code required to build the web page. Click another button and the tool automatically uploads the files to your web-hosting server and you have a complete web page.

At first, tools like these may sound like a great idea, but they have some drawbacks:

- **You have little control over the HTML and CSS code the tools automatically generate.** Because the tools need to generate code for all sorts of environments and applications, the code they generate is somewhat generic and can be bloated and unnecessarily complicated.
- **Because of the code bloat, it's extremely difficult to add or modify any of the code that the tools generate.**
- **When you use a graphical desktop tool to create your website, you're stuck using that tool forever.** Just like other desktop software packages, graphical desktop tools often change features as new versions come out. Old features are dropped and new features are added, sometimes forcing you to change the way you design your website. You're stuck in an endless loop of purchasing upgrades and learning new features just to maintain your website.
- **The WYSIWYG principle isn't always accurate.** The layout you create in the canvas may not always represent what appears in web pages for all browsers and devices that people use to view your website.

## Web-hosting sites

Besides the graphical desktop tools, there are also web-hosting sites that mimic that type of web page design. Web-hosting sites such as Squarespace and Weebly are oriented toward novice non-programmers who want to build their own websites. These sites allow people with no experience to get a simple static website up and running in practically no time, and as you can imagine, they're becoming very popular.

These sites have all the same drawbacks as the desktop graphical tools. Plus, many of them don't even let you see the HTML and CSS code that they generate. With these template-based sites, you're completely at their mercy. You can never migrate your web application to a different host (which is exactly what they want).

### Word processors

Some word-processing software packages, such as Microsoft Word and Apple Pages, offer the ability to convert documents into web pages. This feature has the same drawbacks as the fancier WYSIWYG tools: You can't control the code they generate, and the code they do generate is often bloated. Stay away from creating web pages using word processors.

WARNING

Also stay away from the temptation to write your web application code using a word processor. Most word processors embed binary characters into the text, even if you save the document in a text mode. This causes all sorts of problems when you try to view the web page in a browser.

## Working with the Right Tools

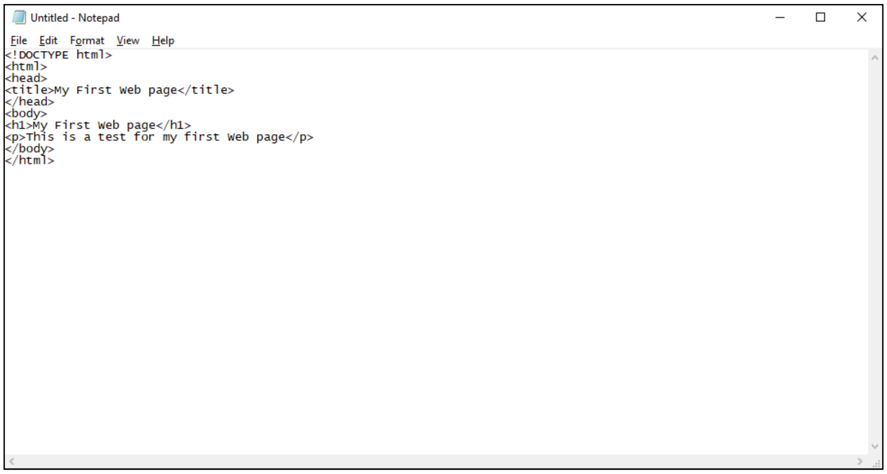
Now that you know which tools to avoid, you're ready to look at the tools you can use to get the job done right. In this section, I fill you in on text editors, program editors, integrated development environments, and browser debuggers.

### Text editors

The hammer-and-saw equivalent for creating web applications is the standard text editor. You can build all the program code used in this book using the text editor that's already installed on your computer. You don't have to buy any fancy software packages or maintain any upgrades. This section explains how to use the standard text editors that are found on most computers, based on the operating system you're running.

#### If you're running Microsoft Windows

If you're running Microsoft Windows, you have the trusty Notepad application for creating and viewing standard text files. Notepad provides a bare-bones interface for typing text and saving it. Figure 3-1 shows an example of writing HTML code in a Notepad window. Notepad is nothing fancy — just your code in black and white.



```
<!DOCTYPE html>
<html>
<head>
<title>My First Web page</title>
</head>
<body>
<h1>My First Web page</h1>
<p>This is a test for my first web page</p>
</body>
</html>
```

**FIGURE 3-1:** Using Microsoft Notepad to write HTML code.

Notepad works fine as a programming tool, but you'll want to tweak a few of the settings before you start coding in Notepad, just to make things easier.

### DISABLING WORD WRAP

In Notepad, you can define the width of the document you want to create, and then Notepad automatically starts a new line when you've reached that limit. This feature is handy for typing memos, but it causes issues when coding.

Wrapping a line of code from one line to the next is generally not allowed in programming languages. All the code for a statement should be on the same line, unless you do some trickery to tell it otherwise.

Another issue with word wrap is that the GoTo option in the Edit menu becomes disabled when word wrap is turned on. Because Notepad doesn't show line numbers, the GoTo feature is all you have to hunt for specific line numbers that error messages point out. GoTo is a crucial tool to have in the Notepad editor.

To disable word wrap in Notepad, click the Format entry in the menu bar; then click the Word Wrap entry to ensure there is no check mark next to it.

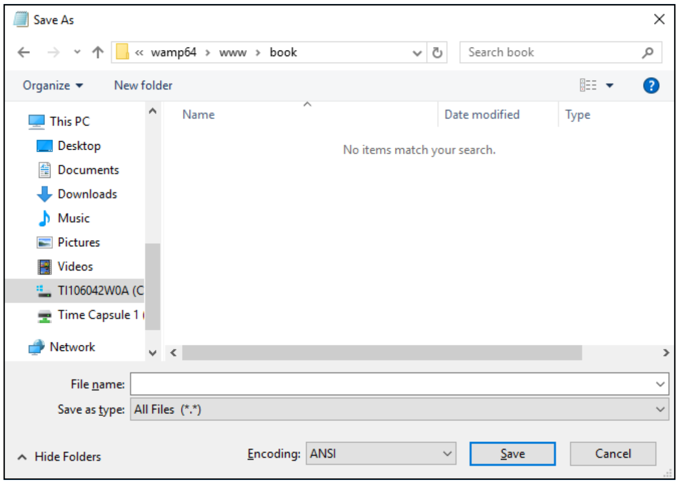
### AVOIDING DEFAULT FILE EXTENSIONS

By default, Notepad assumes you're saving a text document and automatically appends a `.txt` file extension to the file. That doesn't work with programming code, because most programs use a specific file extension to identify themselves (such as `.html` for HTML files or `.php` for PHP files).

When you use the File ➪ Save As menu option in Notepad, you'll need to be careful when saving your program file that the `.txt` file extension doesn't get appended to the end of the filename. To save a program file using Notepad, follow these steps:

1. **Choose File ➪ Save As from the menu bar at the top of the editor.**

   The Save As dialog box, shown in Figure 3-2, appears.



File name:

Save as type: All Files (*.*)

Encoding: ANSI

**FIGURE 3-2:** The Microsoft Notepad Save As dialog box.

2. **In the drop-down list at the top of the Save As dialog box, navigate to the folder where you want to save the program file.**
3. **From the Save As Type text box near the bottom of the Save As dialog box, select All Files (*.*).**

   This prevents Notepad from appending the `.txt` file extension to your filename.
4. **In the File Name field, enter the filename for your program file, including the file extension you want to use.**
5. **Click Save to save the program file.**

   Your program file is properly saved in the correct format, with the correct filename, in the correct location.

## SEEING FILE EXTENSIONS

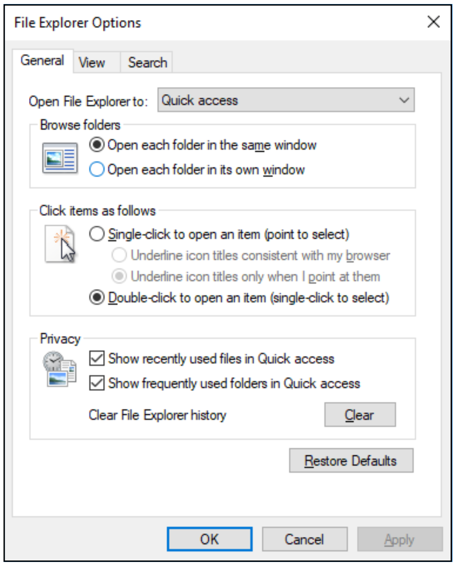
In Microsoft Windows you use File Explorer to navigate the storage devices on your system to open files. Unfortunately, the default setup in File Explorer is to hide the file extension part of the filename (the part after the period) so that it doesn't confuse novice computer users.

That can have the opposite effect for programmers, adding confusion when you're trying to look for a specific file. You may use the same filename for multiple files with different extensions. Fortunately, you can easily change this default setting in Windows. Just follow these steps:

1. **In Windows 8 or 10, open Settings. In Windows 7, open the Control Panel.**
2. **In Windows 8 or 10, type** File Explorer Options **in the search bar and press Enter.**
3. **Click the icon for the File Explorer Options tool that appears in the search results.**
4. **In Windows 7, click the File Explorer Options icon in the Control Panel.**

   You may have to go to the Advanced view to see it.

   After you open File Explorer Options, the dialog box should look like Figure 3-3.



Open File Explorer to: Quick access

Browse folders
● Open each folder in the same window
○ Open each folder in its own window

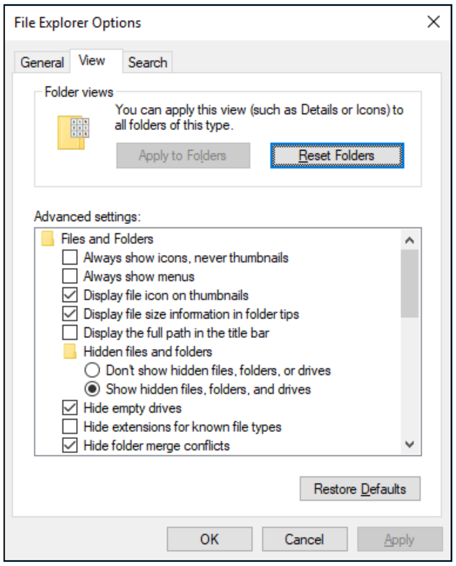
Click items as follows
○ Single-click to open an item (point to select)
○ Underline icon titles consistent with my browser
● Underline icon titles only when I point at them
● Double-click to open an item (single-click to select)

Privacy
☑ Show recently used files in Quick access
☑ Show frequently used folders in Quick access
Clear File Explorer history

**FIGURE 3-3:** The File Explorer Options dialog box in Windows.

5. **Click the View tab.**
6. **Remove the check mark from the Hide Extensions for Known File Types check box, as shown in Figure 3-4.**
7. **Click OK.**

   Now you'll be able to see the full filename, including the extension, when you look for your programs using File Explorer.



You can apply this view (such as Details or Icons) to all folders of this type.

Advanced settings:

- Files and Folders
  - ☐ Always show icons, never thumbnails
  - ☐ Always show menus
  - ☑ Display file icon on thumbnails
  - ☑ Display file size information in folder tips
  - ☐ Display the full path in the title bar
  - Hidden files and folders
    - ○ Don't show hidden files, folders, or drives
    - ● Show hidden files, folders, and drives
  - ☑ Hide empty drives
  - ☐ Hide extensions for known file types
  - ☑ Hide folder merge conflicts

**FIGURE 3-4:** Removing the Hide Extensions for Known File Types check mark.

### SETTING THE DEFAULT APPLICATION

Now that you can see the full filename of your program files in File Explorer, there's just one more hurdle to cross. If you want to open your program files using Notepad by default, you'll need to tell File Explorer to do that. Follow these steps:

1. **Navigate to the program file, and right-click the filename.**
2. **In the menu that appears, select Open.**

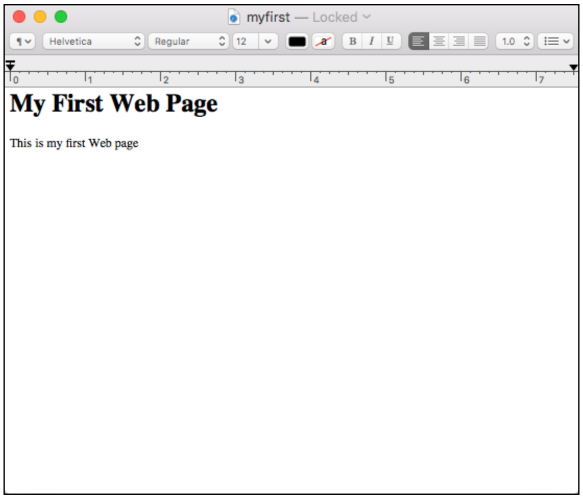
   The Open dialog box appears.
3. **Select Notepad from the list of programs, and then select the check box to always open files of this type using the program.**

   Now you'll be able to double-click your program files in File Explorer to automatically open them in Notepad.

## If you're running macOS

If you're running macOS (or one of the earlier Mac OS X versions), the text editor that comes standard is called TextEdit. The TextEdit application actually provides quite a lot of features for a standard text editor — it recognizes and allows you to edit a few different types of text files, including rich text files (`.rtf`) and HTML files.

The drawback to TextEdit is that sometimes it can be *too* smart. Trying to save and edit an HTML file in TextEdit can be more complicated than it should be. By default, TextEdit will try to display the HTML tags as their graphical equivalents in the editor window, as shown in Figure 3-5.



# My First Web Page

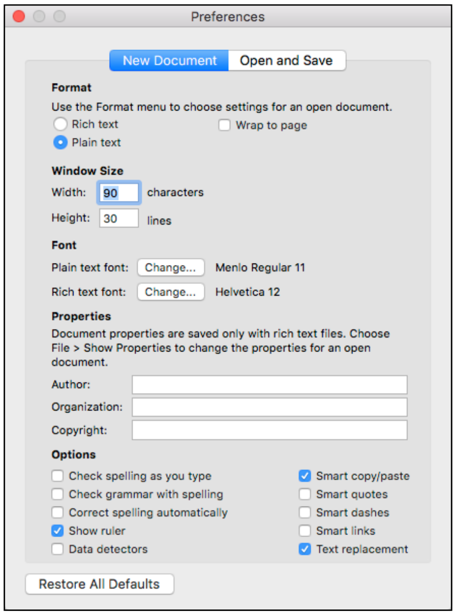
This is my first Web page

**FIGURE 3-5:** Using the default TextEdit settings to edit an HTML file.

As you can see in Figure 3-5, TextEdit actually shows the text as the HTML tags format it instead of the actual HTML code. This won't work for editing an HTML file, because you need to see the code text instead of what the code generates. There's an easy way to fix that — just follow these steps:

1. **Choose TextEdit ➪ Preferences.**

   The Preferences dialog box, shown in Figure 3-6, appears.



**Format**

Use the Format menu to choose settings for an open document.

- Rich text
- [✓ Plain text]
- Wrap to page

**Window Size**

Width: 90 characters

Height: 30 lines

**Font**

Plain text font: Change... Menlo Regular 11

Rich text font: Change... Helvetica 12

**Properties**

Document properties are saved only with rich text files. Choose File > Show Properties to change the properties for an open document.

Author:

Organization:

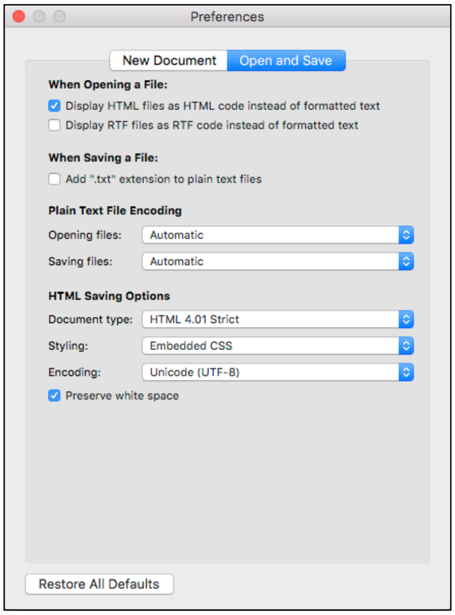
Copyright:

**Options**

- Check spelling as you type
- Check grammar with spelling
- Correct spelling automatically
- ✓ Show ruler
- Data detectors
- ✓ Smart copy/paste
- Smart quotes
- Smart dashes
- Smart links
- ✓ Text replacement

**FIGURE 3-6:** The Preferences dialog box in TextEdit.

2. **On the New Document tab, in the Format section, select the Plain Text radio button.**
3. **In the Options section, remove the check mark from the following check boxes:**
   - Correct Spelling Automatically
   - Smart Quotes
   - Smart Dashes
   - Smart Links
4. **Click the Open and Save tab (see Figure 3-7).**



Preferences

New Document | Open and Save

**When Opening a File:**

☑ Display HTML files as HTML code instead of formatted text

☐ Display RTF files as RTF code instead of formatted text

**When Saving a File:**

☐ Add ".txt" extension to plain text files

**Plain Text File Encoding**

Opening files: Automatic

Saving files: Automatic

**HTML Saving Options**

Document type: HTML 4.01 Strict

Styling: Embedded CSS

Encoding: Unicode (UTF-8)

☑ Preserve white space

Restore All Defaults

**FIGURE 3-7:** The Open and Save tab of the Preferences dialog box.

5. **In the When Opening a File section, check the Display HTML Files as HTML Code Instead of Formatted Text check box.**
6. **In the When Saving a File section, remove the check mark from the Add ".txt" File Extension to Plain Text Files check box.**
7. **Close the Preferences dialog box to save the settings.**

Now you're all set to start editing your program code using TextEdit!

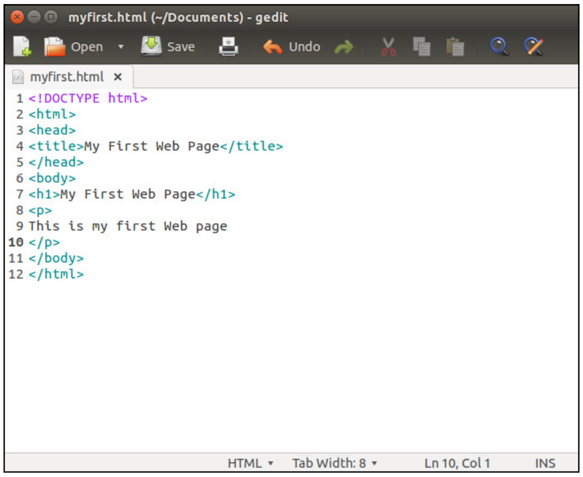
## If you're running Linux

The Linux environment was made by programmers, for programmers. Because of that, even the simple text editors installed by default in Linux distributions provide some basic features that come in handy when coding.

Which text editor comes with your Linux distribution usually depends on the desktop environment. Linux supports many different graphical desktop environments, but the two most common are GNOME and KDE. This section walks through the default text editors found in each.

### THE GNOME EDITOR

If you're working in a GNOME desktop environment, the default text editor is gedit, shown in Figure 3-8.



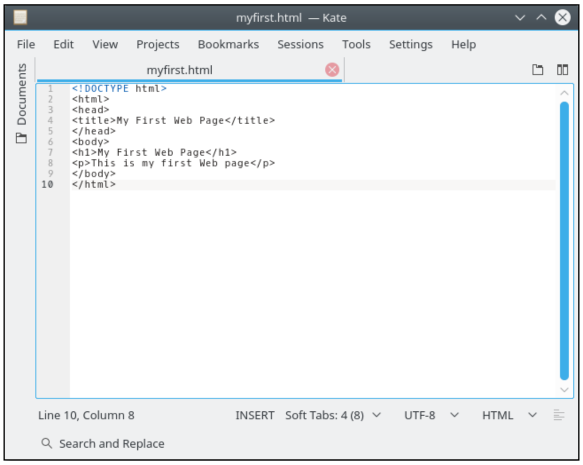
```
<!DOCTYPE html>
<html>
<head>
<title>My First Web Page</title>
</head>
<body>
<h1>My First Web Page</h1>
<p>
This is my first Web page
</p>
</body>
</html>
```

**FIGURE 3-8:** The gedit editor used in Linux GNOME desktops.

The gedit editor automatically saves program files as plain text format and doesn't try to add a `.txt` file extension to filenames. There's nothing special you need to do to dive into coding your programs using gedit. Plus, it has some advanced features specifically for programming that you would find in program editors (see the "Program editors" section later in this chapter).

### THE KDE EDITOR

The default text editor used in the KDE graphical desktop environment is Kate, shown in Figure 3-9.



```
<!DOCTYPE html>
<html>
<head>
<title>My First Web Page</title>
</head>
<body>
<h1>My First Web Page</h1>
<p>This is my first Web page</p>
</body>
</html>
```

**FIGURE 3-9:** The Kate editor used in Linux KDE desktops.

Just like gedit, the Kate editor contains lots of programmer-friendly features right out of the box. Again, no special configuration is required before you can start editing your program code in Kate.

## Program editors

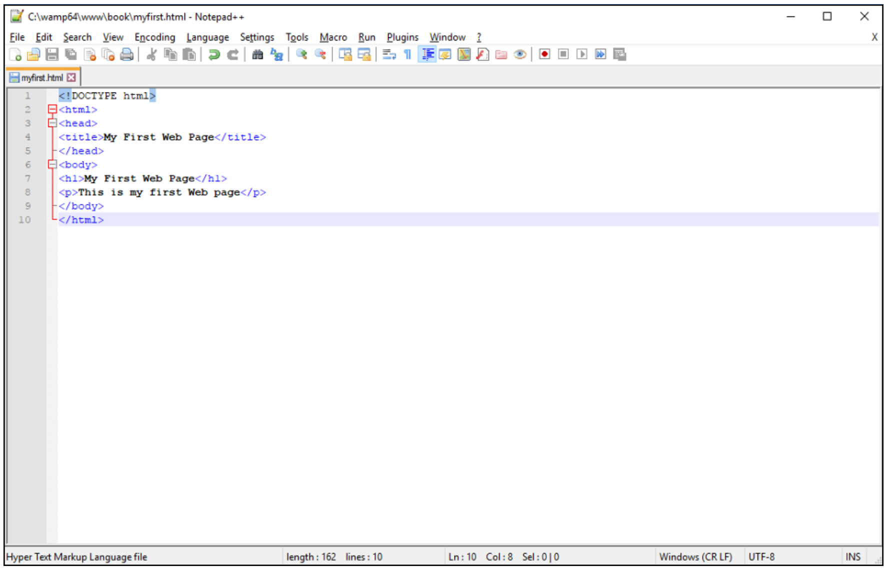
The next step up from standard text editors is a family of tools called *program editors*. A program editor works just like a text editor, but it has a few additional built-in features that come in handy for programming. Here are some of the features that you'll find in program editors:

- **Line numbering:** Providing the line numbers off to the side of the window is a lifesaver when coding. When an error message tells you there's a problem on line 1935, not having to count every line to get there helps!
- **Syntax highlighting:** With syntax highlighting, the editor uses different colors for different parts of the program. Program keywords are displayed using different colors to help make them stand out from data in the code file.
- **Syntax error marking:** Text that appears to be used as a keyword but that isn't found in the code statement dictionary is marked as an error. This feature can be a time-saver by helping you catch simple typos in your program code.

There are lots of commercial program editors, but some of the best program editors are actually free. This section discusses some of the better free ones available for HTML, CSS, JavaScript, and PHP coding.

## Notepad++

If you're running Microsoft Windows, the Notepad++ tool is a great place to start. As its name suggests, it's like Notepad, but better. You can download Notepad++ from `www.notepad-plus-plus.org`. The main editing window is shown in Figure 3-10.


C:\wamp64\www\book\myfirst.html - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
myfirst.html
Hyper Text Markup Language file
length : 162 lines : 10
Ln : 10 Col : 8 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS

```
<!DOCTYPE html>
<html>
<head>
<title>My First Web Page</title>
</head>
<body>
<h1>My First Web Page</h1>
<p>This is my first Web page</p>
</body>
</html>
```

**FIGURE 3-10:** Notepad++.

The main interface for Notepad++ looks similar to Notepad, so there's nothing different to get used to. By default, it shows line numbers along the left margin, as well as the type of file and the column location of the cursor at the bottom.

Notepad++ recognizes the syntax for many different types of programming languages, including HTML, CSS, JavaScript, and PHP. It highlights the keywords and will even match up opening and closing block statements. If you miss a closing block, Notepad++ will point that out.

## Scintilla and SciTE

The Scintilla library (`www.scintilla.org`) is an open-source project to provide a programming text editor engine for use in any type of environment. Developers can embed the Scintilla editor into any type of application free of charge.

The SciTE package is a desktop text editor tool that implements the Scintilla library. The SciTE package is available for Windows, macOS, and Linux platforms. You can download it from the Scintilla website for the Windows and Linux