

Index

Note: **Boldface** numbers indicate code and illustrations; an italic *t* indicates a table.

A

absolute positioning, in HTML, 184–187, **184–187**
abstract classes, 6, **6**
Accept header, 260–265, **261–265**
access control list (ACL)
 ArrayAccess interface in, 89–94, **89–94**
 Iterator interface in, 94–98, **94, 98**
Adapter design pattern, 34–36, **34–36**
Advanced Encryption Standard (AES), 159, 164
Ajax, 189, 197–204, **198–203**
 Document Object Model (DOM) and, 200–204, **201–203**
 JavaScript Object Notation (JSON) in, 197, 199–200, 203
 jQuery in, 198–200, **198–199**
Alexander, Christopher, design patterns, 29–30
aliasing, namespaces, 16–17, **17**
anonymous functions. *See* closures
Apache, and security, 151
APIs
 authentication in, 266–272, **266–272**
 keys and secrets for, 269–272, **269–272**
 version negotiation in, 259–265, **260–265**
ArrayAccess interface, 89–94, **89–94**, 99
ArrayObject class, 98–100
arrays
 PHP vs. JSON, 248–249, **249**

 typing array values in, 101–102, **101–102**
assertion in PHPUnit, 213, **213**
asynchronous processing, 308–315, **309–315**
authentication
 nonsession-based, 268–272, **268–272**
 REST and, 236
 session-based, 266–268, **266–268**
 web services and, 265–272, **266–272**
autoloaders
 namespaces and, 13
 spl_autoload_register() in, 84–86, **84–86**

B

backgrounds, HTML, 178–179, **179**
backslash as namespace separator, 13, 14
basic concepts of PHP, 1–27
BBCode, 140
Bergman, Sebastian, PHPUnit, 206
binding, web services, 241
block vs. inline HTML elements, 168–169, **168**
bootstrap files, unit testing/test-driven development, 210–211, **210, 211**
bootstrap processes, 41
 Front Controller design and, 46
brackets(<>) in HTML, 167
breakpoints, 114, 121–124, **121–124**
browsers, 165–204
 HTML and, 165–187. *See* HTML

Internet Explorer and HTML in, 167
JavaScript and, 187–197. *See also* JavaScript security issues with, 129

C

C++, 66

caching, 296–298, **297**

 pre-, 303

 preprocessing and, 302–308, **304–308**

 web services and, 233

CallbackIterator, 104

calls, in debugging, 115–116, **115–116**

Cascading Style Sheets (CSS), 165, 169,
 174–187, **176–177**

 display attribute in, 181–182, **182**

 float attribute in, 182–183, **182–183**

 home page example without, **175**

 inline, 176

 layouts in, 181–183

 position property in, 183–187, **183–187**

 positioning elements on page, absolute vs.
 relative, 184–187, **184–187**

 selectors in, 179–181

Cipher Block Chaining (CBC), 161–162,
 163

ciphers, 159, 163

classes, 1–4, **2–3**

 abstract, 6, **6**

 Adapter design pattern for, 34–36, **34–36**

 aliasing of namespaces in, 16–17, **17**

 ArrayObject, 98–100

 constants in, 2, 4, **2–3**

 constructors in, 31–32

 dependencies in, 32

 dependency injection/Dependency Injection

 Container (DI Container) in, 74–81,
 74–81, 220–222, **220–222**, 281–285,
 281–285

 Dependency Inversion Principle in, 68–73,
 68–73

 Factory design pattern for, 33, **33–34**

 HTML, 176, 180–181

 Interface Segregation Principle in, 65–68,
 65–68

 Lazy Initialization/Lazy Loading design
 pattern in, 37–40, **38–40**, 102–104,
 103–104

 Liskov Substitution Principle in, 63–65, **64**,
 65

 methods in, 2, **2–3**

 mock objects and, 32–33, 222–229, **223–229**,
 286–289, **287–289**

 OOP programming and, 1–2, **2–3**

 Open-Closed Principle in, 63–64

 properties in, 2, **2–3**

 Single Responsibility Principle (SRP) and,
 59–63, **60–63**, 279

 Singleton design for, 31, **31**

 specification of, in HTML, 176

 Strategy design pattern for, 36–37, **36–37**

 traits and, 17–23, **18–23**

 unit testing/test-driven development and,
 207–209, **207–209**

CLCommand(), 276

CLCommandWithCpf(), 276

CLCommandWithOutput(), 276

click events, in JavaScript, 189–190, **189–190**

CLInteractive(), 276

closures, 23–27, **23–27**

 Closure class and, 25–26, **26**

 defining and using, 23–25, **23–25**

 \$who keyword and, 26–27, **27**

Code Complete (McConnell), 286

collections of data, looping, 84

color, with HTML, 178–179, **179**

conditional statements, 83–84

CONNECT, 234

constants, 4

 in classes, 2, 4, **2–3**

constructors, 31–32

content negotiation, in web services, 264–265,
 264, 265

context, 4

 in classes, 4–5, **5**

controllers, in MVC architecture, 54–55

count() function, Countable interface, 86–89,
 87–89

Countable interface, 86–89, **87–89**, 99

- cross-site request forgery, 131, 142–145, **144**
cross-site scripting, 131, 138–142, **139–142**
CRUD, 70
customization, 40–45
 bootstrap processes and, 41
 design patterns for, 40–45
 Observer design pattern in, 41
 Publish-Subscribe design pattern in, 42
 Visitor design pattern in, 41–42
- D**
dangerous files, security issues, 151–152
data caching, 296–298, **297**
data deletion attacks, 133
Data Description Specifications (DSS), 166
data modification attacks, 133–136, **133–136**
data sources, testing, 222–229, **223–229**
data types, PHP vs. JSON, 248
databases
 cross-site request forgery vs., 142–145, **144**
 cross-site scripting threat in, 138–142, **139–142**
 data deletion attacks to, 133
 data modification attacks to, 133–136, **133–136**
 session fixation vs., 145–146, **145, 146**
 session hijacking vs., 146–147, **147**
 SQL injection security threat in, 131–138, **132–138**
 unit testing/test-driven development and, 226–229, **226–229**
 validating input for, 147–150, **147–150**
DB2, 273
 performance issues and, in connection, 295
Debug URL selection, 111–113, **111–113**
debugging, 105–128. *See also* test-driven development
 breakpoints in, 114, 121–124, **121–124**
 configuring Zend Debugger for, 107, **107**
 Debug URL selection in, 111–113, **111–113**
 expressions in, 120–121, **121**
 firewall restrictions and tunneling in, 107–110, **108–110**
 flow control in, 113–116, **113–116**
 function calls in, 115–116, **115–116**
 host IP addresses for Zend Debugger and, 106
 initiating debug session in, 111–113, **111–113**
 manual control of, 126–128, **127**
 PHPUnit and unit testing in, 218
 ports for Zend Debugger and, 106
 Remove All Terminated Launches in, 114
 Resume in, 114
 Step Into in, 115–116, **115–116**
 Step Over in, 116
 Step Return in, 116
 Suspend in, 114
 Terminate in, 114
 toolbars for, 124–126, **125, 126**
 variables in, 116–120, **116–120**
 XDebug in, 105
 Zend Debugger in, 105–106
decryption, 160–161, **160–161**
DELETE, 133, 234
dependencies
 in classes, 32
 Dependency Inversion Principle in, 68–73, **68–73**
 injection/Dependency Injection Container (DI Container) in, 73–81, **74–81, 220–222, 220–222, 281–285, 281–285**
 dependency injection/Dependency Injection Container (DI Container), 73–81, **74–81, 220–222, 220–222, 281–285, 281–285**
DES ciphers, 159, 163
design patterns, 29–81
 Adapter, 34–36, **34–36**
 customizations using, 40–45
 Factory, 33
 Front Controller, 45–53, **45–53**
 Lazy Initialization/Lazy Loading, 37–49, **38–40, 102–104, 103–104**
 Model/View/Controller (MVC), 53–58, **54–58**
 Observer, 40–45
 Publish-Subscribe, 40–45
 Singleton, 30–33, **30–33**
 SOLID methodology vs., 58–81, 58t

- Strategy, 36–37, **36–37**
- Visitor, 40–45
- DirectoryIterator, 104
- display attribute, 181–182, **182**
- distributed computing, Publish-Subscribe design pattern, 42
- doctype, in HTML, 166–167, **167**
- Document Object Model (DOM), 187, 190–192, **191**, 200–204, **201–203**
- E**
- ECMA, ECMAScript and JSON, 248
- encryption, 152–164, 269–272, **269–272**.
 - See also* hashing
 - Advanced Encryption Standard (AES) for, 159, 164
 - Cipher Block Chaining (CBC) and, 161–162, 163
 - ciphers in, 159, 163
 - decryption and, 160–161, **160–161**
 - DES, Triple DES, AES, Rijndael ciphers in, 159, 164
 - hashing and, 152–158, **153–158**
 - initialization vector (IV) and, 162–163, **162**, **163**
 - mycrypt library for, 159
 - symmetric key, 158–163, **159–163**
- event managers, Publish-Subscribe design pattern, 42–43, **42–43**
- events, JavaScript, 189
- exceptions, in unit testing, 214, **215–216**
- ExecuteProgram(), 276
- export routine, 19, **19**
- expressions, debugging, 120–121, **121**
- F**
- Facebook, 248
- factories, 69
- Factory design pattern, 33, **33–34**
- Fielding, Roy, and HTTP protocol, 235
- filtering
 - security and, 131
 - unit testing/test-driven development and, 214
- firewall restrictions and tunneling, 107–110, **108–110**
- fixation, session, 145–146, **145**, **146**
- float attribute, 182–183, **182–183**
- flow control, in debugging, 113–116, **113–116**
- fonts, with HTML, 178–179, **179**, 181
- foreach, 84, 96
- forgery, cross-site request, 142–145, **144**
- Front Controller design pattern, 45–53, **45–53**
 - building logic into, 49, **49–51**, 52–53
 - mod_rewrite and, 47, **47**
 - Model/View/Controller (MVC) applications and, 46–47, **47**, 55–58, **55–57**
 - rewrite rules using, 46
- function calls, in debugging, 115–116, **115–116**
- function keyword, 188
- functions
 - anonymous. *See* closures
 - closures as, 23–27, **23–27**
 - JavaScript, 188–189, 190
- G**
- GET, 133, 143, 145, 203, 234
- H**
- hashing, 152–158, **153–158**
 - web services and, 269–272, **269–272**
- HEAD, 234
- High Scalability* blog, 298
- hijacking, session, 146–147, **147**
- hinting, type, type hinting, 10–11, **10–11**, 98
- HTML, 165–187
 - <html> node in, 167
 - Ajax and, 197–204, **198–203**
 - backgrounds in, 178–179, **179**
 - block vs. inline elements in, 168–169, **168**
 - brackets (< >) use in, 167
 - Cascading Style Sheets (CSS) in, 169, 174–187, **176–177**. *See also* Cascading Style Sheets (CSS)
 - Class attribute in, 176, 180–181
 - class specification in, 176
 - color in, 178–179, **179**
 - cross-site scripting and, 139–142, **139–142**

- CSS layouts in, 181–183
 - display attribute of CSS in, 181–182, **182**
 - doctype in, 166–167, **167**
 - Document Object Model (DOM) in, 187, 190–192, **191**, 200–204, **201–203**
 - errors and error correction in, 169
 - float attribute of CSS in, 182–183, **182–183**
 - fonts in, 178–**179**, **179**, 181
 - <head> and <body> nodes in, 167
 - hierarchical nature of, 166
 - htmlspecialchars() and htmlentities() functions in, 140
 - hyperlinks in, 170–172, **171**, **172**
 - ID attribute in, 176, 178–179, **179**
 - inline CSS and, 176
 - Internet Explorer use with, 167
 - JavaScript and, 187–197. *See also* JavaScript
 - JavaScript libraries and, 192–197, **192–197**
 - jQuery library in, 192–197, **192–197**
 - lists in, 169–170, **170**
 - navigation lists in, 179–180, **180**
 - node ID specification in, 176
 - nodes in, to form hierarchy of, 167
 - paragraphs in, 169
 - position property in, putting elements in place with, 183–187, **183–187**
 - positioning elements on page, absolute vs. relative, 184–187, **184–187**
 - presentation code separated from, 176
 - security and, 140–142, **140–142**
 - selectors in CSS and, 179–181
 - Standard Generalized Markup Language (SGML) as precursor to, 166
 - Style attribute in, 176
 - tables in, 172–174, **172–173**, **174**
 - tag redefinition in, 178
 - type redefinition in, 176
 - title for documents in, 168, **168**
 - HTML Purifier, 140–142, **140–142**
 - HTTP/HTTPS. *See also* web services
 - Accept header in, 260–265, **261–265**
 - action verbs used with, 234–235, 235*t*, **235**
 - Ajax and, 197–204, **198–203**
 - API version negotiation and, 259–265, **260–265**
 - asynchronous processing and, 308–315, **309–315**
 - authentication in, 266–272, **266–272**
 - caching in, 233
 - content negotiation in, 264–265, **264**, **265**
 - cross-site request forgery vs., 142–145, **144**
 - cross-site scripting and, 139–142, **139–142**
 - htmlspecialchars() and htmlentities() functions in, 140
 - request abstraction in, using SRP and SOLID methodology, 60–63, **61–63**
 - security issues of, 129
 - session fixation vs., 145–146, **145**, **146**
 - session hijacking vs., 146–147, **147**
 - stateless protocols and, 233
 - symmetric key encryption in, 158–164, **159–164**
 - tokens for security in, 143–145, **144**
 - tunneling in, 107–110, **108–110**
 - URL definition in, using REST, 233–234
 - web services and, 231. *See also* web services
 - HTTP Referer, 143
 - hyperlinks, in HTML, 170–172, **171**, **172**
 - Hypertext Markup Language. *See* HTML
- I**
- ID attribute, HTML, 176, 178–**179**, **179**
 - implements keyword, 8
 - indexes, in preprocessing, 299
 - Initialization, Lazy, 37–40, **38–40**, 102–104, **103–104**
 - initialization vector (IV), 162–163, **162**, **163**
 - initiating debug session, 111–113, **111–113**
 - injection/Dependency Injection Container (DI Container), 73–81, **74–81**, 220–222, **220–222**, 281–285, **281–285**
 - inline vs. block HTML elements, 168–169, **168**
 - input validation and security, 131, 147–150, **147–150**
 - instantiation

- dependency injection/Dependency Injection Container (DI Container) in, 75–81, **75–76**
 - Factory design pattern for, 33, **33–34**
 - insteadof operator, 22–23, **22–23**
 - interfaces. *See also* browsers
 - integrated file system (IFS)
 - performance issues and, 294–295, **294, 295**
 - security and, 148
 - Interface Segregation Principle, 65–68, **65–68**, 70
 - interfaces, 6–9, **7–8**
 - ArrayAccess interface in, 89–94, **89–94**
 - Countable, 86–89, **87–89**
 - defining, 6–9, **7–8**
 - Interface Segregation Principle in, 65–68, **65–68**, 70
 - Iterator interface in, 94–98, **94, 98**
 - JavaScript, 195–197, **195–197**
 - predefined, 84–104. *See also* Standard PHP Library (SPL) and
 - internal threats to security, 130
 - Internet Explorer, 167
 - inversion of control, 220. *See also* dependency injection, 220
 - IP addresses, session IDs and session fixation, 145–146, **145, 146**
 - iteration in looping, ArrayObject class, 98–100
 - Iterator interface, 94–98, **94, 98**
 - IteratorAggregate interface, 99
- J**
- Java, 37, 66
 - JavaScript, 165, 187–197
 - Ajax and, 197–204, **198–203**
 - click events in, 189–190, **189–190**
 - cross-site request forgery vs., 143–145, **144**
 - cross-site scripting and, 138–142, **139–142**
 - Document Object Model (DOM) and, 187, 190–192, **191**, 200–204, **201–203**
 - embedding of, 187–188, **187**
 - events in, 189
 - functions in, 188–189, 190
 - iteration or looping in, 194–195, **194–195**
 - JavaScript Object Notation (JSON) in, 197, 199–200, 203, 248–259
 - jQuery in, 198–200, **198–199**
 - jQuery library in, 192–197, **192–197**
 - libraries in, 192–197, **192–197**
 - rendering of, 188
 - session hijacking and, 146–147, **147**
 - user interface with, 195–197, **195–197**
 - variables in, 188–189, **189**
 - writing code in, objects and properties of, 188–189
 - JavaScript Object Notation (JSON), 197, 199–200, 203, 248–259, 264, 309
 - arrays in, 248–249, **249**
 - callback validation in, 257–258
 - class creation in, 258–259, **258, 259**
 - data passing using, 250
 - data types of, 248
 - mapping requests using, 250–251, **251**
 - method definition in, 255–257, **255–257**
 - Remote Procedure Call (RPC) and, 248
 - REST and, 250
 - router creation in, 253–255, **254**
 - service creation using, 251–259, **251–259**
 - SOAP and, 250
 - Toolkit use with, 281
 - validation in, 250
 - XML vs., 250
- job queues, asynchronous processing, 308–315, **309–315**
- JOIN, 301
- jQuery, 198–200, **198–199**
- jQuery library, 192–197, **192–197**
- K**
- keys, API, for web services, 269–272, **269–272**
 - keys, symmetric key encryption, 158–164, **159–164**
- L**
- Lazy Initialization/Lazy Loading, 102–104, **103–104**
 - Lazy Initialization/Lazy Loading design pattern, 37–40, **38–40**

- levels of security, 149, **149**
- libraries, JavaScript, 192–197, **192–197**
- Linux, 294, 295
 - security issues and, 148
- Liskov Substitution Principle, 63–65, **64, 65**
- Liskov, Barbara, 64
- Loading, Lazy, 38–40, **39–40**, 102–104, **103–104**
- looping, 83–84
 - ArrayAccess interface in, 89–94, **89–94**, 99
 - ArrayObject class and, 98–100, 98
 - collections of data and, 84
 - Countable interface and, 86–89, **87–89**, 99
 - foreach and, 84, 96
 - iteration in, ArrayObject class for, 98–100
 - Iterator interface in, 94–98, **94, 98**
 - IteratorAggregate interface in, 99
 - JavaScript, 194–195, **194–195**
 - predefined interfaces and, 84
 - Serializable interface in, 99
 - Standard PHP Library (SPL) and, 84
 - Traversable interface in, 99
 - type hinting and, 98
- Lynx, 165
- M**
- Magento, 285–286, 299
- Markdown, 140
- Martin, Robert C., SOLID methodology, 58–59, 66
- McConnell, Steve, 286
- MD5, 145, 153–158
- memoization, 302–303
- method definition, using JSON, 255–257, **255–257**
- methods, 3
 - in classes, 2, **2–3**
- Microsoft Windows, 294
- mock objects, 32–33, 222–229, **223–229**, 286–289, **287–289**
- mod_rewrite and Front Controller, 47, **47**
- Model/View/Controller (MVC) applications, 53–58, **54–58**
 - architecture of, **54**
 - controllers in, 54–55
 - Front Controller and, 46–47, **47**
 - Front Controller design pattern and, 55–58, **55–57**
 - models in, 54
 - separation of concerns in, 53
 - URL mapping in, 55
 - views in, 55
- models, in MVC architecture, 54
- Mogull, Rich, 130
- mycrypt library, 159
- MySQL, 295, 298
- N**
- namespaces, 11–17, **12–17**
 - aliasing for duplicate class names and, 16–17, **17**
 - autoloaders and, 13
 - backslash as separator in, 13, 14, **14**
 - defining, 12, **12**
 - instantiating, 12, **12**
 - multiple different, working with, 14, **14**
 - use keyword to declare, 15–16, **16**
 - Zend Engine and, 12
- navigation lists, HTML, 179–180, **180**
- NIST, 159, 164
- nodes, in HTML, 167
 - ID specification in, 176
- nonsession-based authentication, 268–272, **268–272**
- O**
- object oriented programming (OOP), 1–27
 - abstract classes in, 6, **6**
 - classes in, 1–2, **2–3**
 - closures in, 23–27, **23–27**
 - context in, 4–5, **5**
 - interface definition in, 6–9, **7–8**
 - namespaces in, 11–17, **12–17**
 - objects in, 1–2, **2–3**
 - polymorphism in, 9, **9**
 - traits in, 17–23, **18–23**
 - type hinting in, 10–11, **10–11**, 98
 - visibility levels in, 5, **5**

objects
 ArrayObject class and, 98–100
 mock, 32–33, 222–229, **223–229**, 286–289, **287–289**
 OOP programming and, 1–2, **2–3**
Observer design pattern, 40–45
ODBC, 273
Open-Closed Principle, 63–64
OPTIONS, 234
outside threats to security, 130–131

P

passwords, hashing, 152–158, **153–158**
Pattern Language, A (Alexander), 30
performance, 293–315
 asynchronous processing and, 308–315, **309–315**
 caching and, 296–298, **297**, 302–308, **304–308**
 DB2 connections and, 295
 integrated file system (IFS) and, 294–295, **294**, **295**
 Linux and, 294, 295
 Microsoft Windows and, 294
 preprocessing and, 298–308
 recent improvements to, from IBM, 293
 Toolkit and, 295
 Zend Server Job Queue and, 308–309
PgmCall(), 276
PHP library. *See* Standard PHP Library (SPL)
PHPUnit, 206–218. *See also* test-driven development
polymorphism, 9, **9**
position property, 183–187, **183–187**
POST, 143, 203, 234
predefined interfaces, 84–104. *See also* Standard PHP Library (SPL) and
predictable locations for security attacks, 151–152
prepare statements, 136–138, **136–138**
preprocessing, 298–308
 caching and, 302–308, **304–308**
 normal calculations and, 298–302, **299–302**
private visibility level, 5, **5**

procedural programming, 1
properties, 3
 in classes, 2, **2–3**
protected visibility level, 5, **5**
public visibility level, 5, **5**
Publish/Subscribe (PubSub) design pattern, 40–45
 class definitions for, 43–44, **44**
 event manager for, 42–45, **42–45**
 subscriber for, 44–45, **44–45**
PUT, 234

Q

qshellCommand(), 276
queries
 cross-site request forgery vs., 142–145, **144**
 jQuery library in, 192–197, **192–197**
 prepare statements in, for security, 136–138, **136–138**
 preprocessing and, 298–308
 session fixation vs., 145–146, **145**, **146**
 session hijacking vs., 146–147, **147**
 SQL injection security threat in, 131–138, **132–138**
queues, asynchronous processing, 308–315, **309–315**

R

registries, 69
relative positioning, in HTML, 184–187, **184–187**
Remote Procedure Call (RPC), 248
Remove All Terminated Launches, in debugging, 114
Representational State Transfer (REST), 232–236, 235*t*, **235**, 309
 architecture of, 232–233
 authentication in, 236
 caching in, 233
 HTTP verbs in, 234–235, 235*t*, **235**
 JSON and, 250
 resource/URL definition in, 233–234
 stateless protocols and, 233

- request abstraction, using SRP and SOLID methodology, 60–63, **61–63**
- REST. *See* Representational State Transfer (REST)
- Resume, in debugging, 114
- Rijndael ciphers, 159, 164
- router creation, using JSON, 253–255, **254**
- RSS, 42
- Ruby on Rails, 53
- ## S
- “script kiddies,” 130
- security. *See also* encryption, 129–164, 161
 browsers and, 129
 cross-site request forgery vs., 131, 142–145, **144**
 cross-site scripting vs., 131, 138–142, **139–142**
 dangerous files and, 151–152
 data deletion vs., 133
 data modification attacks vs., 133–136, **133–136**
 encryption and, 152–164
 filtering output for, 131
 hashing for, 152–158, **153–158**
 HTML and, 140–142, **140–142**
 htmlentities() function in, 140
 htmlspecialchars() function in, 140
 HTTP issues in, 129
 integrated file system (IFS) systems and, 148
 internal threats to, 130
 levels of, 149, **149**
 Linux systems and, 148
 outside threats to, 130–131
 predictable locations for attacks against, 151–152
 prepare statements in, 136–138, **136–138**
 “script kiddie” and, 130
 session fixation vs., 131, 145–146, **145, 146**
 session hijacking vs., 131, 146–147, **147**
 SQL injection vs., 131–138, **132–138**
 storage practices vs., 151–152
 symmetric key encryption in, 158–164, **159–164**
 tokens in, 143–145, **144**
 validating input for, 131, 147–150, **147–150**
 vulnerability of applications and, 131
- SELECT, 133, 298, 301
- selectors, CSS, 179–181
- self keyword, 5
- Serializable interface, 99
- session fixation, 131, 145–146, **145, 146**
- session hijacking, 131, 146–147, **147**
- session-based authentication, 266–268, **266–268**
- sessions, Singleton design, 31–32
- SHA-1/2/...256, etc., 145, 153–158, 269–270
- Simple Object Access Protocol (SOAP), 231, 236–240, **237–240**
 JSON and, 250
 Web Services Definition Language (WSDL) and, 236–238, 240–248, **241–248**
 XML documents and, 236
- Single Responsibility Principle (SRP), SOLID, 59, 281
- Singletons, 30–33, **30–33**, 274
- SOAP. *See* Simple Object Access Protocol (SOAP)
- SOLID methodology, 58–81, 58*t*, 148, 281, 285
 dependency injection/Dependency Injection Container (DI Container) in, 73–81, **74–81**, 220–222, **220–222**, 281–285, **281–285**
 Dependency Inversion Principle in, 68–73, **68–73**
 Interface Segregation Principle in, 65–68, **65–68**, 70
 Liskov Substitution Principle in, 63–65, **64, 65**
 Open-Closed Principle in, 63–64
 Single Responsibility Principle (SRP), 59–63, **60–63**, 279
- spl_autoload_register(), 84–86, **84–86**
- SplPriorityQueue, 104
- SQL injection security threat, 131–138, **132–138**
- Standard Generalized Markup Language (SGML), 166

Standard PHP Library (SPL), 83–104
 ArrayAccess interface in, 89–94, **89–94**, 99
 ArrayObject class and, 98–100
 Countable interface in, 86–89, **87–89**, 99
 Iterator interface in, 94–98, **94**, **98**
 IteratorAggregate interface in, 99
 Serializable interface in, 99
 spl_autoload_register() in, 84–86, **84–86**
 Traversable interface in, 99
 type hinting and, 98
stateless protocols, 233
Step Into, in debugging, 115–116, **115–116**
Step Over, in debugging, 116
Step Return, in debugging, 116
storage practices vs. security, 151–152
Strategy design pattern, 36–37, **36–37**
Style attribute, HTML, 176
Suspend, in debugging, 114
symmetric key encryption, 158–164, **159–164**

T

tables, in HTML, 172–174, **172–173**, **174**
tabling, 302–303
tags, redefining, in HTML, 178
Terminate, in debugging, 114
test suites, 219–220, **219**, **220**
test-driven development, 205–230. *See also*
 debugging
 assertion in PHPUnit and, 213, **213**
 bootstrap files for, 210–211, **210**, **211**
 classes and class structure in, 207–209,
 207–209
 data sources and, 222–229, **223–229**
 database connections and, 226–229, **226–229**
 debugging and, 218
 dependency injection and, 220–222, **220–222**
 exception throwing in, 214, **215–216**
 failure in, 213–214, **213**, 216, **216**
 filtering in, 214
 mock objects and, 222–229, **223–229**
 output of PHPUnit in, 213, **213**
 passing code, 216, **216–217**, 218, 229, **229**
 PHPUnit in, 206–218
 PHPUnit menu in, 212, **212**

 PHPUnit test case window in, **207**
 repeating the test in PHPUnit, using F11, 218
 selecting test file for, 212, **212**
 test suites in, 219–220, **219**, **220**
 Toolkit and, 285–291, **286–291**
 unit testing in, 206, 285–291, **286–291**
Textile, 140
\$this keyword, 5
timestamp, 15, **15**
title, in HTML, 168, **168**
tokens, in security, 143–145, **144**
toolbars, debugging, 124–126, **125**, **126**
Toolkit, 273–292
 abstraction for, 273–274
 calling a program in, 276
 calls into, 276, **276**
 CL command execution in, 276, **276**
 connections and InternalKey parameter
 setting in, 275, **275**
 dependency injection with, 281–285,
 281–285
 encapsulating individual calls into classes
 for, 278–280, **279–280**
 error checking/testing in, 289–291, **289–291**
 instantiating ToolkitService class in, 274, **274**
 mock objects in, 287–288, **287**, **288**
 parameter creation and setting for, 276–277,
 276–277
 performance issues and, 295
 source code for, 273
 testing inline functionality with, 278
 ToolkitService class in, 274–277, **274–277**
 unit testing with, 285–291, **286–291**
 XML and, 273
ToolkitService class, 274–277, **274–277**
TRACE, 234
traits, 17–23, **18–23**
 adding to class, 19, **19**
 collisions in, 22–23, **22–23**
 defining, 18, **18**
 insteadof operator and, 22–23, **22–23**
 testing for use of, 20–22, **20–22**
Traversable interface, 99
Triple DES cipher, 159, 163

- tunneling, 107–110, **108–110**
 Twitter, 248, 298
 type hinting, 10–11, **10–11**, 98
 types
 array values, 101–102, **101–102**
 redefining, in HTML, 176
 validating, 101–102, **101–102**
- U**
 unit testing, 206. *See also* test-driven
 development
 Toolkit and, 285–291, **286–291**
 URL mapping, in MVC implementations, 55
 URLs and web services, using REST, 233–234
 use keyword, namespaces declaration, 15–16,
 16
 user interface, JavaScript, 195–197, **195–197**
- V**
 validation
 JSON, 250
 security and, input, 131, 147–150, **147–150**
 validators, 101–102, **101–102**
 typing array values and, 101–102, **101–102**
 var keyword, 188
 variables
 debugging, 116–120, **116–120**
 Interface Segregation Principle in, 66–68
 JavaScript, 188–189, **189**
 version negotiation, API, for web services,
 259–265, **260–265**
 views, in MVC architecture, 55
 visibility levels, 5, **5**
 Visitor design pattern, 40–45
- W**
 web services, 231–272
 Accept header in, 260–265, **261–265**
 adding and deleting input parameters for,
 using WSDL, 242–243, **243**
 API version negotiation in, 259–265,
 260–265
 authentication in, 266–272, **266–272**
 authentication in, using REST, 236
 binding content for, using WSDL, 244–246,
 246
 binding creation for, using WSDL, 241, **241**
 caching in, 233
 callback validation in, using JSON, 257–258
 class creation in, using JSON, 258–259, **258**,
 259
 content negotiation in, 264–265, **264**, **265**
 encryption in, 269–272, **269–272**
 endpoint setting for, using WSDL, 246–248,
 247, **248**
 hashing and, 269–272, **269–272**
 HTTP and, 231
 HTTP action verbs and, 234–235, 235*t*, **235**
 JSON and, 264
 key and secret for APIs in, 269–272,
 269–272
 mapping requests in, using JSON, 250–251,
 251
 method definition in, using JSON, 255–257,
 255–257
 new service creation in, using JSON,
 251–259, **251–259**
 new service creation in, using WSDL,
 240–241, **241**
 nonsession-based authentication in, 268–272,
 268–272
 port type defined for, using WSDL, 242, **242**,
 246–247, **247**
 Remote Procedure Call (RPC) and, using
 JSON, 248
 Representational State Transfer (REST) in,
 232–236, 235*t*, **235**
 return type and browser for, using WSDL,
 243–244, **244**, **245**
 router creation in, using JSON, 253–255,
 254
 session-based authentication in, 266–268,
 266–268
 setting operation name for, using WSDL,
 243, **243**
 Simple Object Access Protocol (SOAP) and,
 231, 236–240, **237–240**
 stateless protocols and, 233

- Web Services Definition Language (WSDL)
 - and, 236–238, 240–248, **241–248**
 - X-based headers in, 259–260
- Web Services Definition Language (WSDL),
 - 236–238, 240–248, **241–248**
 - adding and deleting input parameters in, 242–243, **243**
 - binding content setting in, 244–246, **246**
 - binding creation in, 241, **241**
 - creating files in, 240
 - creating new service in, 240–241, **241**
 - endpoint setting in, 246–248, **247, 248**
 - port type definition in, 242, **242**, 246–247, **247**
 - return type and browser setting in, 243–244, **244, 245**
 - setting operation name in, 243, **243**
- WHERE, 133
- \$who keyword, 26–27, **27**
- Wikitex, 140
- WordPress, 53
- WSDL. *See* Web Services Definition Language (WSDL)
- X**
- X-based headers, in web services, 259–260
- XDebug, 105
- XML
 - JSON vs., 250
 - SOAP and, 236–240, 237–**240**
 - Toolkit use and, 273, 281
- Y**
- YAML, Toolkit use, 281
- Z**
- Zend Debugger, 105–106. *See also* debugging
 - breakpoints in, 121–124, **121–124**
 - configuration of, 107, **107**
 - Debug URL selection in, 111–113, **111–113**
 - expressions in, 120–121, **121**
 - firewall restrictions and tunneling in, 107–110, **108–110**
 - flow control in, 113–116, **113–116**
 - host IP addresses for, 106
 - initiating debug session in, 111–113, **111–113**
 - manual control of, 126–128, **127**
 - ports for, 106
 - toolbars for, 124–126, **125, 126**
 - variables in, 116–120, **116–120**
- Zend Engine, 104
 - ArrayAccess interface in, 89–94, **89–94**
 - namespaces and, 12
- Zend Framework, 60
- Zend Server, 106
- Zend Server Job Queue, 308–309
- Zend Studio, 106, 206, 240