abstract classes, 186–190
            defining with abstract keyword, 186–187
            final keyword in, 188
            hasAccess() method and, 189
            methods as, 189
            this keyword in, 188
            type hinting and, 188
            abstract keyword, 186–187
access type constants, Zend Core, 293, 293
Active Record, 326
adding data to tables, 237–240, 237–240
Adobe, 334
Agile development, 161
AIX, 219
Akismet, 335
Amazon, 335
AND, 41, 42
Apache, 7, 219, 335
            frameworks for, 321–322
applications for PHP, 3–4
arguments, func_get_args() and, 126–127
array_keys() function, 103
array_merge() and array_merge_recursive() functions, 107–110
array_rand() function, 104–105
array_shift() function, 106
array_walk() in, 123
arrays, 13, 15, 79–112
            accessing items in, 81
            appending data to, 82
            association operator with, 87
            associative, 79, 84–90
            associative sort of, using asort()/arsort() functions, 99–100
            callback function and, array_walk() in, 123
            calling parameters in, with
            call_user_func_array(), 125–126
            count elements in, using count() function, 94
            creating, using array construct, 80–84
            data typing and, 89
            declaration of, 90
            error handling in, 92–93
            explode() function in, 305
            extract values from, using extract() function, 104
            i5_fetch_array() function for, 294
            input validation for, using in_array() function, 103
            iterating through, 83–84, 89–90
            key definition for, 85–88
            key retrieval from, using array_keys() function, 103
            key sorts of, using ksort()/krsort() functions, 101
            looping and, 83–84, 89–90

NOTE: Boldface indicates illustrations and code; t indicates a table.
arrays, continued
  merging, array_merge() and array_merge_recursive() functions, 107–110
  multidimensional, 90–95
  natural order sort of, using natsort() function, 101–102
  numerical, 79
  pointers and, next(), current(), reset() and, 100–101
  quotation marks within, 86
  random values from, using array_rand() function, 104–105
  range of, using range() function, 95
  reduce elements in, using unset() function, 105–106
  references and, 83–84
  referencing items in, 91–92
  remove first element in, using array_shift() function, 106
  reverse sort of, using rsort(), 98–99
  size of, and sizeof() function, 94
  sizing of, 92
  sorting, with sort() function, 95–102, 96
  var_dump() function in, 305
  zero-basis of, 81
asort()/arsort() functions, 99–100
assignment operator, 31–32
association operators, 87
associative arrays. See arrays, associative
audio streams, 157–158
Audioscrobbler, 335
authority, in MySQL, 226
auto_increment keyword, 250
autoloader() function, 211–213
autoloading data into classes, 163, 211–213

B
break keyword, 61
break statement, 62, 73–75
backslash (/) escape character, 17, 29–30
basename(), 148
Berkeley Software Distribution (BSD), 332, 334
bitwise shift operator, 115
boolean data type, 13, 14
braodcast() function, 211–213
abstract classes, 186–190
  defining with abstract keyword, 186–187
  final keyword in, 188
  hasAccess() method and, 189
  methods as, 189
  this keyword in, 188
  type hinting and, 188
  adding data to, 165
  autoloading data into, 163, 211–213
  callback in, 213–215
  class keyword and, 161
  constructors and, 167–168
  contexts in, 178–180
  curly braces ({ }) used in naming, 161
  defining, 161–162
  destructors and, 169–172
  exceptions and, 206–213
  Exception class (Exception::) and methods for, 208
  C
C language, 275
Cake PHP framework, 330, 331
call(), 201, 205–206
call_user_func(), 124–125
call_user_func_array(), 125–126
callbacks, 122–123, 213–215
calls. See also program calls, 290
calling array parameters in, with call_user_func_array(), 125–126
calling functions, call_user_func() for, 124–125
canonicalized file names and realpath(), 148
Cascading Style Sheets (CSS), 2
  model-view-controller (MVC) framework and, 320
case statements, 61, 62
case-sensitivity in PHP, 251
casting, 13–14
catch(), 211, 208–209
CHAIN, 291
changeIt(), 46–47
CL programs, calling PHP script from, using QSH command, 290–291
class keyword, 161
classes, 159–218
  abstract classes, 186–190
    defining with abstract keyword, 186–187
    final keyword in, 188
    hasAccess() method and, 189
    methods as, 189
    this keyword in, 188
    type hinting and, 188
  adding data to, 165
  autoloading data into, 163, 211–213
  callback in, 213–215
  class keyword and, 161
  constructors and, 167–168
  contexts in, 178–180
  curly braces ({ }) used in naming, 161
  defining, 161–162
  destructors and, 169–172
  exceptions and, 206–213
  Exception class (Exception::) and methods for, 208
getCode(), 208
getFile(), 208
getLine(), 208
getMessage(), 208
getTrace(), 208, 209
getTraceAsString(), 208, 209
new keyword to throw, 210–211
throw() in, 208–209
try()/catch() blocks in, 208–209, 211
type checking and, 206–208
uses for, 209–210
extends keyword and, 174–175
functions attached to, 165. See also methods
getInstance() method in, 179
inheritance in, 173–178
instance of object in, 163
instantiation, 163
interfaces and, 180–184
implements keyword and, 181–182
interface keyword for, 180–181
type hinting and, 184
magic methods for, 167, 196–206
call(), 201, 205–206
get(), 201–202
isset(), 201, 203–204
serialization in, using sleep and wakeup, 196–200
set(), 201, 202–203
string manipulation in, using toString, 200–201
unset(), 201, 204
manipulating objects in, 164
methods and, 165–166
naming of, 161
naming Private/Protected methods in, 191–192
object orientation (OO) concepts and, 159–161
benefits of, 159–160
data integrity and, 160
databases and, 161
re-use and re-instantiation of objects and, 160
request-response actions and, 160
overloading members and methods in, 201–206
parameter passing in, 168–172
parent keyword and, 177–178
Password Change method example for, 173
permissions (Private, Public, Protected) in, 190–196, 190r
polymorphism and, 184–186
properties of, 162
re-use and re-instantiation of objects and, 160
residence of, 160
self keyword and, 180
separating process from data in, 169
serialization in, using sleep and wakeup, 196–200
static keyword and, 180
this keyword and, 166, 180
type hinting and, 184, 185–186, 188
unset function in, 170
visibility keyword and, 172–173
Zend Core and, 276–277
zvals in, 170–171
client–side applications, 6, 7
closedir(), 145–146
closing MySQL, mysql_close() function for, 234
closing tag for PHP (?>), 19
closing tag, 25–26
Cobol, 275
code examples in book, 22
Codeligniter framework, 332, 333, 334
collections in DB2, 246, 246–247
comments, 15–16
PHPDocumenter and, 16, 309–312, 310r, 312
community for PHP, 4
comparison operators, 37–40
compound data types, 13
compression streams, 155–156
concatenation () operator, 34, 36–37
conditional statements, 26–27, 57–63
curly brace ({} delimiters in, 27–28, 60
if statement as, 57–60
switch statement as, 60–63
construct(), 196
constructors, 167–168
contexts, 178–180
continue keyword in loops, 69–73, 74
control structures, 57
curly brace ({} delimiters in, 27–28

NOTE: Boldface indicates illustrations and code; t indicates a table.
Controller class, in model-view-controller (MVC) framework, 322, 323
cookies, 261–266
creating (baking), 263–265, 263t, 264
debugging, 314
properties of, 262–263, 262
reading, using reference, 264
security and, 265–266, 266
set with HTTP header() function, 263
set with setcookie() function, 263, 263
/COPY compiler directive, 51
COUNT() function, arrays and, 94
CREATE TABLE statements, MySQL, 237
create_function(), 124
creating databases/tables in DB2, 247–253, 247–253
CSS. See Cascading Style Sheets
curly braces ({}), 27
class naming and, 161
conditional statements and, 60
variable names and, 17, 29
current() function, 100–101

D
Data Access Objects (DAOs), 240–245,
241–243, 244, 254–256, 254–256
data access vs. data handling in PHP, 138
data integrity, 160
data types, 12–15
arrays and, 15, 89. See also arrays, 15
casting, 13–14
changing at runtime, 13
System i program call example and, constants for, 281, 282

type checking, exceptions, and, 206–208
type hinting and, 184, 185–186, 188
weak typing of, 13
data wrapper, 156–157, 156t, 157t
database access, 3, 161, 219–259
adding records in, 298–303, 299–302
adding records in, i5_addnew(), i5_setvaluet() in, 300
case-sensitivity in PHP, 251
closing server connection with mysql_close() function in, 234
collections in DB2 and, 246, 246–247
connecting to server with mysql_connect() function in, 233
creating, 234–240, 235–236
Data Access Objects (DAOs) and, 240–245,
241–243, 244, 254–256, 254–256
collection handling with mysql_error() function in, 234
extensions and, 226, 227–229, 245, 257
extensions of PHP for (IBM_DB2), 5
functions for, 233–234
IBM_DB2, 245–256
connecting to using db2_connect() function, 245–247, 246
creating databases/tables in, 247–253,
247–253
error handling in, using
db2_conn_errormsg() and
db2_stmt_error(), 248
extensions for, 245, 257
field information using var_dump() function in, 247
functions for, 245–247
generated keyword in, 250
PHP Data Objects (PDOs), 256–258, 257
queries in, using db2_exec() function, 248
system tables in, 247, 248
LAMP development standards and, 219
MySQL, 219
auto_increment keyword and, 250
command-line access to, 225
granting authority in, 226
host systems in, 225
installation and setup of, 219–229
mysqlcheck command to verify installation of, 223–224, 224
password for, 221–222, 222
Portable Application Solution Environment (PASE) and, 221, 224
setup for, 224–229
starting server for, 222–224
user names in, 225
mysql_create_db() and, 235
naming conventions for, 226
PHP Data Objects (PDOs) and, 227, 256–258, 257
prepared statements and, 227
queries using mysql_query() function in, 233, 234
retrieving rows using mysql_fetch() function in, 234
tables in, 229–245
   adding data to, 237–240, 237–240
   CREATE TABLE statements for, 237
   creating, 234–240, 235–243
   Data Access Objects (DAOs) and, 240–245, 241–243, 244, 254–256
functions to operate on, 233–234
libraries/files vs., 229–230, 230–236
   rows (records) in, 229–230, 230–231
   SELECT command in, 230–233, 231–232
   value objects (VOs) and, 240–245, 241–243, 244, 254–256
Zend Core and, record-level file access in, 291–303
db2_conn_errormsg(), 248
db2_connect() function, 245–247, 246
db2_exec() function, 248
db2_stmt_error(), 248
do-while loop, 68
documentation using PHPDoc, 16
Dojo, 335
dollar sign ($)variable name indicator, 12
DOMDocument, 26
double colon (::) operator, 180
DOW/ENDDO, 67
DSPSRVPGM CL command, 285–286
dynamic vs. static content, 6

e-mail
QSH command and, 291
QTMMSENDMAIL API and, 291
echo statement, 29, 32
Eclipse, 307–308. See also Zend Studio for Eclipse
EDI, 326
else statement, 63
elseif statements, 60
equality operators, 37–39
error handling
arrays and, 92–93
Exception class (Exception::) and methods for, 208
exceptions and, 206–213
IBM_DB2 and, using db2_conn_errormsg() and db2_stmt_error(), 248
MONITOR/ENDMON (RPG) functions, 208
mysql_error() function for, 234
try/throw() blocks in, 208–209
Zend debugger for, 313–316

escape characters/sequences, 17–19, 18t, 29–30
exam and exercise solutions/answers, 22
Exception class (Exception::) and methods, 208
exceptions, 206–213
Zend debugger for, 313–316

NOTE: Boldface indicates illustrations and code; t indicates a table.
exceptions, continued
getFile(), 208
getLine(), 208
getMessage(), 208
getTrace(), 208, 209
getTraceAsString(), 208, 209
new keyword to throw, 210
– 211
throw() in, 208 – 209
try/catch() blocks in, 208 – 209, 211
type checking and, 206 – 208
uses for, 209 – 210
executing a script, 21
existence testing, file_exists(), 144
existence testing, function_exists(), 128 – 129
explode() function, 305
extends keyword, 174 – 175
Extensible Markup Language. See XML
extensions
database access and, 226, 227 – 229
IBM_DB2 and, 245, 257
save handler extensions for, 268 – 270, 268 – 270
sessions and, 268
extract() function, 104

F
fclose(), 147
feof(), 142t, 143
fetch data elements (i5_fetch_xxx()), 303
i5_fetch_object() function for, 294
i5_fetch_assoc() function for, 294
i5_fetch_row() function for, 294
into arrays in, i5_fetch_array() function for, 294
fget(), 141t
fgetcsv(), 141t
fgets(), 141t, 143
field information using var_dump() function, 247
field naming, System i program call example
and, 282, 283
file management, 133 – 158
close with fclose(), 147
common functions for, 146 – 148
create directory for, with mkdir(), 147
data access vs. data handling in, 138
directory handling and, 19, 145 – 146. See also
directories
end of, with feof(), 142t, 143
existence checking with file_exists(), 144
file name retrieval only with basename(), 148
“file” functions for, 144 – 145
gen single character in, with fget(), 141t
gen single line in, with fgets(), 141t, 143
include statement and, 134
include_once statement and, 134
include_path statement and, 134 – 135
indexed file access in PHP for, 295 – 296, 295 – 296
lock/unlock, with flock(), 142t
locking access in, 293, 293
meta-information and, 139, 139t
metadata and, 137
multiple directories and, include_path
with, 135
New file in, 19
open file using i5_open() function for, 293
open, with fopen(), 140 – 141, 147
permissions and, 138
read permissions for, with is_readable(), 147
read single CSV line from, with fgetcsv(), 141t
read specified bytes from, with fread(), 141t,
142 – 143
read with file(), 144
read with file_get_contents(), 144, 145
reading data in, functions for, 141 – 144, 141t
record-number file access in PHP for,
296 – 298
relative path names and, 134 – 135
require statement and, 134
require_once statement and, 134
resource variables and, 138 – 139
saving files in, 19
scripting language and, 133 – 134
See k file in, with fseek(), 140
sequential file access in PHP for, 292 – 294,
292, 293, 293t, 294t
size of, very large files and, 144
size of, with filesize(), 144
streaming resources and, 138 – 139
temporary file creation, with tmpfile(), 147 – 148
unlink with unlink(), 147
wrappers and, 139–149, 148–158
  audio stream, 157–158
  compression streams, 155–156
  data, 156–157, 156t, 157t
  filesystem, 148–150, 149–150t
  FTP/FTPS, 151–152, 152t
  glob stream, 157–158
  HTTP/HTTPS, 150–151, 150t, 151t
  process interaction stream, 157–158
  SSH2, 157
write permissions for, with is_writable(), 147
file name retrieval with basename(), 148
file(), 144
file_exists(), 144
file_get_contents(), 144, 145
files vs. tables, 229–230, 230t
filesize(), 144
filesystem wrapper, 148–150, 149–150t
final keyword, 188
find data in, i5_
  See k() functions for, 296, 303
Firebug, 8, 335
fixation, session, 271
Flash, 334
Flickr, 335
float data type, 13
fopen(), 140–141, 147
for loops, 64–67
FOR/ENDFOR operators, 64
foreach loop, 83–84, 89–90
Fox Interactive, 334
frameworks, 319–339
  Apache rules in, 321–322
  application directory structure for, 321, 321t
  application layout in, 320, 321
  Cake PHP, 330, 331
  CodeIgniter, 332, 333, 334
  Controller class in, 322, 323
  model-view-controller (MVC) in, 319–320
  overview of, 326–339
  Symfony as, 327, 328, 329
  View class in, 324–325, 324
fread(), 141t, 142–143
f. See k(), 140
FTP/FTPS wrapper, 151–152, 152t
func_get_args(), 126–127
function functions. See functions
function_exists(), 128–129
functions, 113–131
  arguments in, with func_get_args(), 126–127
  arrays and, array_walk() in, 123
  callbacks and, 122–123
  calling array parameters in, with
    call_user_func_array(), 125–126
  calling, with call_user_func(), 124–125
  classes and, 165. See also method
  creating, using create_function(), 124
  exceptions and, 206–213
  existence testing for, with function_exists(), 128–129
  function keyword and, 113
  function type, 122–130
  GET parameter and, 124
  global variables accessed from within, 46–48
  hashing, 124–125
  IBM_DB2 and, 245–247
  internal, 113
  lambda, 124
  main(), 119–120
  MySQL and, 233
  output buffering, with ob_start(), 129
  parameters in, 115–116
    optional, 117–118, 117
    overloading of, 117–118, 117
    passing by reference, 118–119
    passing, 116
  passing connections in, 120
  request end and,
    register_shutdown_function(), 129–130
  return value of, using return keyword, 114–115
  scope of, global, 49
  user-defined, 113–119
  variable scope and, 119–122
  variable type, 122
  variables, global and, 120–122

**NOTE:** Boldface indicates illustrations and code; t indicates a table.
G
generated keyword, 250
GET, 67, 124, 265
get(), 201–202
getCode(), 208
getFile(), 208
getinstance() method, 179
getLine(), 208
getMessage(), 208
getTrace(), 208, 209
getTraceAsString(), 208, 209
glob streams, 157–158
global keyword, 47–48
global variables, 46–48, 120–122
super–, 21, 48, 121
Google, 334, 335
granting authority, in MySQL, 226
Gutmans, Andi, 5, 307

H
hasAccess() method, 189
hash functions, 124–125, 176–177
calling array parameters in, with
call_user_func_array(), 125–126
md5(), 125, 176–177
sha1(), 125
header() function and cookie setting, 263
request in, 261–262
security and, 265–266, 266
sessions and, 261, 266–273. See also sessions
Zend for Eclipse and, 314
HTTP/HTTPS wrapper, 150–151, 150t, 151t
Hypertext Markup Language. See HTML

I
i5_addnew(), 300
i5_command() function, 304–305, 304
i5_connect() function, 281
i5_data(See k) function for, 296–298
i5_fetch object() function, 294
i5_fetch_array() function, 294
i5_fetch_assoc() function, 294
i5_fetch_row() function for, 294
i5_fetch_xxx() functions, 303
i5_open() function, 293
i5_program_call() function, 281, 282, 283
i5_program_prepare() function, 281, 282, 285, 289
i5_program_prepare_PCML() function for, 283
i5_See k() functions, 296, 303
i5_setvalue(), 300, 301
i5_update() function, 300, 302–303, 302–303
IBM, 3, 5, 307
IBM_DB2, 5, 245–256
collections in, 246, 246–247
connecting to using db2_connect() function,
245–247, 246
creating databases/tables in, 247–253, 247–253
error handling in, using db2_com_errormsg() and db2_stmnt_error(), 248
extensions for, 245, 257
field information using var_dump() function in, 247
functions for, 245–247
generated keyword in, 250
PHP Data Objects (PDOs), 256–258, 257
queries in, using db2_exec() function, 248
system tables in, 247, 248

htmlspecialchars() function, 272
HTML
compression streams and, 155–156
cookies and, 261–266. See also cookies
header() function and cookie setting, 263
request in, 261–262
security and, 265–266, 266
sessions and, 261, 266–273. See also sessions
Zend for Eclipse and, 314
HTTP/HTTPS wrapper, 150–151, 150t, 151t
Hypertext Markup Language. See HTML
Zend Core record-level file access in, 291–303
if statement, 57–60, 63, 69
curly brace ({} delimiters in, 28–29
if-else statements, 59–60
IGN, 334
images, 6
implements keyword, 181–182
in_array() function, 103
include files, 51–55
variables, variable scope and, 53–55
include statement, 51–55, 134
include_once, 51–55, 134
include_path, 55, 134–135
increment/decrement operators, 34–36
/INCLUDE compiler directive, 51
indexed file access in PHP, 295–296, 295–296
Indianapolis Motor Speedway, 334
infinite loops, 65, 66–67
inheritance, 173–178
extends keyword and, 174–175
parent keyword and, 177–178
initializing a session, using session_start() function, 266–267
input validation, in_array() function for, 103
installation
installing MySQL, 219–229. See also database access
installing PHP IDE, 8
installing Zend Core for i5/OS, 8–11
instance of objects in class, 163
instantiation, getinstance() method in, 179
integers, 12
arrays for, 79
integrated development environments (IDEs), 7–8, 307
interface keyword, 180–181
interfaces, 180–184
implements keyword and, 181–182
interface keyword for, 180–181
type hinting and, 184
internal functions. See functions
Internet Information Services (IIS), 3
interpolation, variable, 17, 29
is_readable(), 147
is_writable(), 147
isdir(), 146
isset(), 93, 201, 203–204
ITER, 69, 70
iterating a loop, 66, 69–75

J
JavaScript, 2, 7, 335
model-view-controller (MVC) framework and, 320
JavaScript Object Notation (JSON), 6
JQuery, 335

K
keys
arrays and, 85–88
arrays and, key retrieval from, using array_keys() function, 103
merging arrays and, array_merge() and array_merge_recursive() functions for, 107–110
pointers and, next(), current(), reset() and, 100–101
sorts of, using ksort()/krsort() functions, 101
ksort()/krsort() functions, 101

L
lambda functions and create_function(), 124
LAMP development standards, 2, 219
language elements, 25–30
LEAVE, 72, 73, 74
LEFT OUTER SQL query, 68
Lerdor, Rasmus, 5
libraries vs. tables, 229–230, 230
linking files, unlink(), 147
Linux, 219, 291
literals, 29
local variables, 49
locking, 293, 293
flock(), 142
logical operators, 41–43
looping, 26–27, 57, 64–75
arrays and, 89–90

NOTE: Boldface indicates illustrations and code; t indicates a table.
Index

looping, continued
arrays and, 83–84
break keyword in, 73–75
continue keyword in, 69–73, 74
curly brace ({ }) delimiters in, 27–28
do-while, 68
for, 64–67, 64
foreach, 83–84, 89–90
infinite, 65, 66–67
iteration for, 66, 69–75
post-ops and, 64
while, 67
Lucene, 335

M
magic methods, 167, 196–206
call(), 201, 205–206
construct(), 196
destruct(), 196
get(), 201–202
isset(), 201, 203–204
overloading members and methods and,
201–206
PHP Data Objects (PDOs) and, 196
serialization in, using sleep and wakeup,
196–200
set(), 201, 202–203
string manipulation in, toString, 200–201
unset(), 201, 204
main() function, 119–120
math operators, 32–34
md5() hash function, 125, 176–177
memory_limit setting, 144
merging arrays, array_merge() and
array_merge_recursive() functions for, 107–110
meta-information, 139, 139t
metadata, 137
methods, 165–166
abstract, 189
callback in, 213–215
classes and, 165–166
magic, 167, 196–206
naming Private/Protected, 191–192
overloading, magic methods for, 201–206
Password Change, 173
PHP Data Objects (PDOs) and, 196
serialization in, using sleep and wakeup,
196–200
string manipulation using toString, 200–201
visibility keyword and, 172–173
Microsoft, 3, 335
MIME, QTMMSENDMAIL API and, 291
mkdir(), 147
model-view-controller (MVC), 190, 319–320.
See also frameworks
MONITOR/ENDMON (RPG) functions, 208
Mozilla, 330
multidimensional arrays. See arrays,
multidimensional
MySQL, 3–4, 219. See also database access
auto_increment keyword and, 250
closing server connection with mysql_close()
function in, 234
command-line access to, 225
CREATE TABLE statements in, 237
creating databases and tables using, 234–240,
235–236
error handling with mysql_error() function
in, 234
extensions for, enabling, 227–229
functions in, 233–234
granting authority in, 226
host systems in, 225
installation and setup of, 219–229
mysql_connect() function in, 233
mysql_create_db() and, 235
mysqlcheck command to verify installation
of, 223–224, 224
password for, 221–222, 222
Portable Application Solution Environment
(PASE) and, 221, 224
queries using mysql_query() function in,
233, 234
recordsets/query results from, 230–233,
231–232, 233
retrieving rows using mysql_fetch() function
in, 234
security issues for, 226

350
SELECT command in, 229, 230–233, 231–232, 233
setup for, 224–229
starting server for, 222–224
table access in, 229–245
UPDATE command in, 229
user names in, 225
mysql_close() function, 234
mysql_connect() function, 233
mysql_create_db(), 235
mysql_error() function for, 234
mysql_fetch() function, 234
mysql_query() function, 233
creating databases/tables using, 235–236
mysqlcheck command to verify installation, 223–224

N
naming conventions
for classes, 161
for database access, 226
for Private/Protected methods, 191–192, 191
for sessions, session_name() function, 267
natsort() function, 101–102
natural order sort, using natsort() function, 101–102
Netscape, 261
new keyword to throw exceptions, 210–211
newline character (/n), 30
next() function, 100–101
Nirvanix, 335
NOT operator, 143
null data type, 13
numbers, sorting of, 97–98
numerical arrays. See arrays, numerical

O
ob_start(), 129
object data type, 13
object orientation (OO) concepts, 116, 159–161
  benefits of, 159–160
  callback in, 213–215
  data integrity and, 160
  databases and, 161
  exceptions and, 206–213
  frameworks in, 319–339
  inheritance and, 173–178
  polymorphism and, 184–186
  re-use and re-instantiation of objects and, 160
  request-response actions and, 160
  serialization in, using sleep and wakeup, 196–200
Onion Store, The, 330
open file using i5_open() function, 293
opendir(), 145–146
opening tag for PHP (<?php>, 19
opening tag, 25–26
operators, 30–46, 30–31, 180
  assignment, 31–32
  basic math, 32–34
  bitwise shift, 115
  comparison, 37–40
  concatenation, 34, 36–37
  equality, 37–39
  increment/decrement, 34–36
  logical, 41–43
  precedence of, 44–46, 44t
  reference, 50–51
  ternary, 43–44
  value comparison, 40
OR, 41, 42
OTHER, 63
output buffering, ob_start(), 129
overloading members and methods, 201–206
overloading parameters, 117–118

P
paamayim nekudotayim (::), 180
parameter passing parameters, 115–116
  i5_program_prepare() function to pass, 282, 285, 289
  optional, 117–118
  overloading of, 117–118
  passing by reference, 118–119
  passing, 116, 168–172
  zvals and, 170–172, 170
parent keyword, 177–178
passing connections, 120
Password Change method example, 173
passwords, MySQL and, 221–222, 222
path names, relative, 134–135
PDF files, 6
PDO_IBM, 5
Perl, 2
permissions, 138
  Private, Public, Protected, 190–196, 190t
Personal Home Page/Forms Interpreter. See
PHP/FI
PHP Data Objects (PDOs), 227, 256–258, 257
  magic methods and, 196
php.ini, 55, 134
  memory_limit setting in, 144
PHP/FI, 5
PHPDocumenter, 16, 309–312, 310t, 312
  type hinting and, 185–186, 188
Portable Application Solution Environment
(PASE), MySQL and, 221, 224
Portable Document Format. See
PDF files
POST, 21, 153, 265, 315
post-decrement, 35
post-ops and looping, 64
POSTwrappers, _POST, 21
pre-decrement, 35
precedence of operators, 44–46, 44t
prepared statements, database access and, 227
print_r(), 313
Private visibility/permission, 190–196, 190t
process interaction streams, 157–158
profiler, Zend for Eclipse, 313–316
Program Call Markup Language (PCML),
288–290, 289–290
program calls
  calling PHP script from RPG or CL using
    QSH command, 290–291
i5_progam_call() function for, 281, 282, 283
Program Call Markup Language (PCML) in,
288–290, 289–290
System i program call example in, 279–285,
280–281
Zend Core and, 275–276
Program Status Data Structure (PSDS), 208
  properties of a class, 162
Protected visibility/permission, 190–196, 190t
Public visibility/permission, 190–196, 190t
Python, 2
Q
QSH/QShell command, 290–291
QTMMSENDMAIL API, 291
queries
  db2_exec() function for, 248
  mysql_query() function in, 233, 234
  recordsets/query results from, 230–233,
    231–232, 233
SELECT command in, 230–233, 231–232, 233
quotation marks, 16–17, 29
arrays and, 86
ecape characters/sequences using, 17–19, 18t
R
random values from arrays, array_rand()
  function, 104–105
range() function, arrays and, 95
re-use and re-instantiation of objects in PHP, 160
READ, 291, 303
read functions, 141–144, 141t
read permissions, is_readable(), 147
read type constants, Zend Core and, 294, 294t
readdir(), 145–146
realpath(), 148
ReCaptcha, 335
record-level file access, Zend Core and, 291–303
record-number file access in PHP, 296–298
records
  adding, i5_addnew(), i5_setvalue() in, 300
  adding, 298–303, 299–302
  updating, 298, 302–303, 302–303
  updating, i5_update() function in, 300,
    302–303, 302–303
recordsets, 230–233, 231–232, 233
retrieving rows using mysql_fetch() function
  in, 234
reduce array elements, unset() function, 105–106
reference operator, 50–51
references, 50–51
  arrays and, 83–84, 91–92
  cookies and, reading from, 264
  parameter passing by, 118–119
  this variable and, 166
  zero-basis of, in PHP, 81
  zvals and, 170–172
register_shutdown_function(), 129–130
relative path names, 134–135
remove first array element, array_shift() function, 106
request end, register_shutdown_function(), 129–130
request-response actions, 160
require statement, 51–55, 134
require_once statement, 51–55, 134
reset() function, 100–101
resource data type, 13
resource variables, 138–139
RETURN, 114
System i program call example and, 286–288, 286–287
return keyword, 114–115
rows (records) in tables, 229–230, 230
retrieval functions in, Zend Core, 294, 294
retrieving, using mysql_fetch() function in, 234
RPG, 1–2, 3, 275
  calling PHP script from, using QSH command, 290–291
  error handling in, 208
  Zend Core program examples for, 277–290
rsort() function, 98–99, 122
S
save handler extensions, 268–270, 268–270
scope of variables, 46–50, 53–55, 119–122
  include files and, 53–55
scripting language, 5, 133
  client–vs. server–side, 6, 7
Secure Shell (SSH), 157
security
  cookies and, 265–266, 266
  fixation of sessions and, 271
  FTP/FTPS wrapper and, 151–152, 152
  HTTP and, 265–266, 266
HTTP/HTTPS wrapper and, 150–151, 150t, 151t
  include files and, include and require statements in, 135–136
MySQL databases and, 226
  sessions and, 270–273
SSH2 wrapper, 157
  visibility/permissions(Private, Public, Protected) in, 190–196, 190
SELECT command, MySQL, 229, 230–233, 231–232, 233
  See also queries SELECT/WHEN, 61–63
self keyword, 180
semicolon (;) statement delimiter, 19, 26–27
  sequential file access, Zend Core and, 292–294, 292, 293, 294
serializing, using sleep and wakeup, 196–200
server–side applications, 6, 7
service program function, System i program call example and, 285–286
session ID, 266
  session_regenerate_id() function for, 271–273
$_SESSION superglobal variable for, 266
session_start() function, 266–267
  session_id() function in, 267
set(), 201, 202–203
setcookie() function, 263, 263
  SETLL, 291, 303
setup for MySQL, 224–229
sha1() hash function, 125

NOTE: Boldface indicates illustrations and code; t indicates a table.
short tags, 26
SimpleXML, 26
Simpys, 335
single vs. double quotes, 16–17
singleton design pattern, 179, 190, 194, 195
size of file with file size(), 144
sizeof() function, arrays and, 94
sleep method, 196–200
SlideShare, 335
Smarty templating engine, 8
Soap, 326
sort() function, 95–102, 96t, 122
sorting
arrays and, 95–102
associative, using asort()/arsort(), 99–100
keys, using ksort()/ksort() functions, 101
natural order, using natsort() function, 101–102
numbers as strings, 97–98
pointers and, next(), current(), reset() and, 100–101
reverse order, using rsort(), 98–99
rsort(), 122
sort() function for, 95–102, 96t, 122
strings of numbers, 97–98
source files, including., 51–55. See also include files
Source Force, 334
special characters, htmlspecialchars() function for, 272
squareIt() function, 48–50
SSH File Transfer Protocol (SFTP), 157
SSH2 wrapper, 157
static context, 178–180
static keyword, 180
STATIC keyword, 49
static variables, 49
static vs. dynamic content, 6
streaming resources, 138–139
Strikrلون, 335
string data type, 12
strings
arrays for, 79
concatenation operator and, 36–37
quotation marks in, 16–17, 29
sorting of, 97–98
toString() magic method in, 200–201
variable interpolation and, 17, 29
strpos() function, 207
Sun Microsystems, 219
superglobal variables, 21, 48, 121
$_SESSION superglobal variable for, 266
Suraski, Zeev, 5, 307
switch keyword, 61
switch statement, 60–63
Symfony framework, 327, 328, 329
syntax of PHP, 11–19, 25–30
SYSTABLES, 247, 248
System i and PHP, 1–4, 5
System i commands with Zend Core,
i5_command() function for, 304–305, 304
System i program call example, 279–285,
280–281
calling using i5_program_call() function for, 281, 282, 283
data type constants for, 281, 282, 283
d�数PSRPGM CL command in, 285–286
field naming in, 282, 283
i5_connect() function for, 281
parameter setting using i5_program_prepare() function for, 281, 282, 285, 289
parameter setting/return values using i5_program_prepare_PCML() function for, 283
Program Call Markup Language (PCML) in, 288–289, 289–290
service program function for, 285–286
TESTAPI2 and, 280
Tookit_classes.php functions example for, 283, 284–285
usage type constants for, 281–282, 281t
values RETURNed from procedures in, 286–288, 286–287
system tables, 247, 248

T
tables, 229–245
accessing, 229–245
adding data to, 237–240, 237–240
CREATE TABLE statements for, 237
tables, continued
  creating, using mysql_query() function, 235–236
  Data Access Objects (DAOs) and, 240–245,
    241–243, 244, 254–256, 254–256
  recordsets/query results from, 230–233,
    231–232, 233
  rows (records) in, 229–230, 230
  SELECT command in, 230–233, 231–232,
    233
  system tables (IBM), 247, 248
  value objects (VOs) and, 240–245, 241–243,
    244, 254–256, 254–256
Technorati, 335
templates, Smarty templating engine and, 8
temporary file creation, with tmpfile(), 147–148
ternary operator, 43–44
TESTAPI service program, 277, 277–278
TESTAPI2 service program, 278, 278–279
  System i program call example and, 280
this keyword, 180, 188
this variable, 166
throw(), 208–209
tmpfile(), 147–148
Tockit_classes.php functions, System i program
  call example, 283, 284–285
toString(), 200–201
TRUE/FALSE, 13, 32, 38, 39, 41–44, 143, 233
try()/catch() blocks, 208–209, 211
tunneling, Zend for Eclipse and, 314–316,
  315–316
Twitter, 335
type hinting. See hinting, type

U
underscore, for naming Private/Protected
  methods, 191–192
unset() function, 105–106, 170, 201, 204
UPDATE command, MySQL, 229, 291
usage type constants, System i program call
  example and, 281–282, 281t
user interfaces, 1–2
user names, in MySQL, 225
user-defined functions. See functions,
  user-defined

V
  value comparison operators, 40
  value objects (VOs), 240–245, 241–243, 244,
    254–256
  var_dump(), 214, 247, 305, 313
  var_export(), 313
  variable functions, 122
  variables
    casting types of, 13–14
    changeIt() function and, 46–47
    changing data types of, 13
    curly braces ({}) and names of, 17, 29
    data types and, 13–15
    dollar sign ($) indicator for, 12
    extract values to, using extract() function, 104
    global, 21, 46–48, 120–122
    include files, scope, and, 53–55
    interpolation and, 17, 29
    isset() function and, 93
    local, 49
    names for, 12, 12t
    POST and, 21
    quotation marks and, 16–17
    references to, 50–51
    resource, 138–139
    scope of, 46–50, 119–122
    static, 49
    superglobal, 21, 48, 121
    weak typing of, 13
  View class, in model-view-controller (MVC)
    framework, 324–325, 324
  visibility keyword, 172–173
  visibility/permissions (Private, Public, Protected)
    in, 190–196, 190t

W
  wakeup method, 196–200
  weak typing in PHP, 13
  WebSphere Development Studio, 307
  while, 34, 67
  Wildfire, 335
  Windows, 291
  wrappers, 139–140, 148–158
  audio stream, 157–158

NOTE: Boldface indicates illustrations and code; t indicates a table.
Index

wrappers, continued
  compression streams, 155–156
data, 156–157, 156r, 157t
filesystem, 148–150, 149–150t
FTP/FTPS, 151–152, 152t
glob stream, 157–158
HTTP/HTTPS, 150–151, 150t, 151t
process interaction stream, 157–158
SSH2, 157
write permissions, is_writable(), 147
WSDL, 2

X
XML, 2, 6, 26
XOR, 42, 43

Y
Yahoo, 327, 335
Yale, 330

Z
Zend Core for i5/OS, 4, 5, 7–8, 257–306, 307
access type constants in, 293, 293r
adding records in, 298–303, 299–302
adding records in, i5_addnew(), i5_setvalue() in, 300
administration startup/login in, 10–11, 11
Apache server startup in, 9
arrays and, explode() function in, 305
arrays and, var_dump() function in, 305
calling PHP script from RPG or CL using QSH command, 290–291
classes in toolkit of, 276–277
fetch data elements in, 294–303
  i5_fetch_object() function for, 294
  i5_fetch_assoc() function for, 294
  i5_fetch_row() function for, 294
  i5_fetch_xxx() functions, 303
  i5_fetch_array() function for, 294
find data in, i5_See k() functions for, 296, 303
  i5_connect() function for, 281
i5_program_call() function for, 281, 282, 283
indexed file access in, 295–296, 295–296
locking access in, 293, 293
main menu screen, 9, 9
open file using i5_open() function for, 293
parameter setting using i5_program_prepare() function for, 281, 282, 285, 289
program calls to/from PHP and, 275–276
read type constants in, 294, 294t
record-level file access in, 291–303
record-number file access in, i5_data_See k() function for, 296–298
row retrieval functions in, 294, 294t
RPG programs for examples used in, 277–290
sequential file access in, 292–294, 292, 293, 293t, 294t
Services Management menu in, 9–10, 10
startup of, 9
subsystem startup in, 10
System i commands with, i5_command() function for, 304–305, 304
System i program call example in, 279–285, 280–281
  calling using i5_program_call() function for, 281, 282, 283
data type constants for, 281, 282
DSPSRVPGM CL command in, 285–286
field naming in, 282, 283
i5_connect() function for, 281
parameter setting using i5_program_prepare() function for, 281, 282, 285, 289
parameter setting/return values using i5_program_prepare_PCML() function for, 283
Program Call Markup Language (PCML) in, 288–290, 289–290
service program function for, 285–286
TESTAPI2 and, 280
Toolkit_classes.php functions example for, 283, 284–285
usage type constants for, 281–282, 281t
values RETURNed from procedures in, 286–288, 286–287
TESTAPI service program in, 277, 277–278
TESTAPI2 service program in, 278, 278–279

356
testing installation of, 8–11
updating records in, 298, 302–303, 302–303
updating records in, i5_update() function in, 300, 302–303, 302–303
Zend Studio for Eclipse (ZSE) /i5/OS, 307–317, 336
bridge code assist of, 308, 309
debugger in, 313–316
downloading, 317
Eclipse development and, 307–308
event details report from, 317, 317
integration with i5 and, 308–309
PHPDocumenter and, 309–312, 310r, 312
profiler in, 313–316
Toolkit code assist of, 308, 308
tunneling in, 314–316, 315
Zend Framework integration and, 312, 313
Zend platform integration in, 316–317
Zend Studio Toolbar in, 315, 316
Zend Platform, 307
Zend Studio. See Zend Studio for Eclipse
Zend Studio Toolbar, 315, 316
Zend Support Forums, 8
Zend Technologies, 3, 5, 307
zero-basis of arrays in PHP, 81
zvals, 170–171

NOTE: Boldface indicates illustrations and code; t indicates a table.