

Index

A

`_a` flag, `rsync`, 564

abstract class diagram, observer pattern, 84–86

abstract classes

designing widgets to descend from, 88

indicating inheritance from, 38

interfaces vs., 77

in PHP 6, 29

when to use in class diagram, 77

abstract methods

`DataBoundObject` class, 181

defining in parent class to implement in child, 174, 181

implementing all interface, 26

implementing in subclasses, 26

`AbstractInstrument` class, 77–81

academic skills, of PHP professional, 630

accessor methods

`DataBoundObject` class, 183

overview of, 10–11

prototyping business objects, 160

account manager, choosing, 411

activation, UML sequence diagrams, 45

activity diagrams, UML, 42–44, 190

actors, use case diagrams, 33–35

`AddConstraint()` method, 329

`_addFunction` method, `SoapServer`, 229

`addItem()` method, `Collection` class, 105–107, 109

`addObserver()` method, `Observable` interface, 86, 87–88

`addRecord()` method, `Observable` interface, 88

administrator

CMS building, 588–589

login to Mantis as, 547–548

Advanced PHP Debugger (APD), 664

agenda, identifying project, 395

aggregates, UML class diagrams, 39

agile development methodologies, 426

***Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process* (Ambler and Jeffries), 630**

algorithms

performance bottlenecks in, 664

performance problems, 662

Alternative PHP Cache (APC), 662

ampersand (&), 81

annotating bugs, Mantis, 553

Apache Web server

choosing for PHP installation, 674

configuration in live staging environment, 557, 558

configuration in studio staging environment, 557

handling PHP high traffic/high availability, 598–599

installing PHP and, 677–679

testing installation, 679

Ulysses only compatible with, 362

APC (Alternative PHP Cache), 662

APD (Advanced PHP Debugger), 664

app log, 209

application architecture

MVC. See MVC (model-view-controller) design pattern

overview of, 305–306

true templating, 333–342

unit testing. See unit testing

application frameworks

application structure, 356

database abstraction, 359

deployment considerations, 379–380

development environments for. See development environments

example application. See Ulysses

form persistence, 358

object relation mapping, 358–359

overview of, 355–356

Prado, 377–378

application frameworks (continued)

- separation of code and display logic, 356–357
- summary, 380
- URL rewriting, 357
- use case scenario, 359–361
- utility classes, 359
- validation, 357
- Zend Framework, 378–379

application servers, 599, 616

application structure

- frameworks providing, 356
- Ulysses framework, 376

architecture, high-performance, 667

archive mode, rsync, 564

`$arRelationMap` member variable, 170

arrows, Zend Studio Debug window, 651

ASP-based Web sites, Web servers for, 673

asset servers

- hardware load balancing and, 604
- improving infrastructure performance, 615

associations, class diagrams, 37–38

associative arrays (hashes), Smarty, 337

attribute types, class diagrams, 36

authentication

- HTTP, 277
- sessions and. See sessions and authentication
- SMS code-sending as, 250

author, CMS building, 588

automated version control, repository extraction.

See also version control

- overview of, 562
- using CVS, 562–563
- using subversion, 563

availability, in PHP. See high traffic/high availability

B

backup requirements, 434

bandwidth

- calculating CIR, 437
- calculating monthly transfer, 438
- high, 595

base64 encoding, 531–533

BaseCamp, 428

basic authentication, live staging, 558

`beginTransaction()` method, PDO, 142–143

bi-directional navigability, 37

binary files, version control and, 646

`bindColumn()` method, PDOStatement, 147

binding element, WSDL, 236

`bindParam()` method, PDO, 140–141

`bindValue()` method, PDO, 141

bitwise logic, MVC mini toolkit constants, 312–314

black box

- creating test suite for your class, 345
- unit testing with, 540

blank servlets, 366

bleeding servers, 424

blind recipients, 260, 262

bookmarks, 280

Boolean flags, 170

bottlenecks, performance

- methods causing, 666–667
- tracking down, 663–664
- types of, 659–661

branching, version control technique, 645

breakpoints, Zend debugging, 653

brief, formal project

- budget, 401–403
- business requirements, 398–399
- commercial terms, 403–404
- future plans, 404
- look and feel, 404–405
- overview of, 397–398
- scope, 399–400
- summary, 405–406
- support, 405
- technology, 405
- timelines, 400–401
- when to say no, 408–409

broadcast e-mail providers, 274

budget

- business requirements, 434
- receiving formal brief of, 401–403

bugs, in Mantis, 550–554

build process, 424

builder pattern

- considerations, 100
- Director, 99–100
- implementation of, 97–99
- overview of, 96–97

building application

- cleaning up, 475–482
- finalized Travel Expense Report, 514–529
- finishing iteration, 483–492
- mock objects, 529–533

starting project. See starting project, building application
 starting the work. See starting work, building application
 summary, 533
 Travel Expense Report, 492–514

bulk e-mails, 274**business analysis, object modeling, 344****business objects**

adding data binding, 160–162
 constructing working version of, 166–168
 design, 157
 lazy instantiation for, 173
 limitations of, 172
 meeting requirements, 166–169
 overview of, 156–157
 property monitoring, 173
 prototyping, 158–160
 relation mapping, 162
 reusability, 173
 in Ulysses, 363–365
 understanding, 169–171
 utility classes vs., 343

business requirements

benefits of OOP, 4
 determining, 433–434
 formal brief of, 398–399

C**caching**

high-performance and, 667–668
 to maximize infrastructure power, 613–614

`_call` **method**, SoapClient, **228**

`call_user_func()`, **callbacks, 111–115**

callbacks, 110–115

implementing, 113–115
 overview of, 110–111
 using `call_user_func`, 111–113

capturing output, Smarty, 342

carbon copy (Cc:) recipients, e-mail, 260–262, 267

career

as a PHP professional. See PHP, professional career in
 as professional software developer, 628–631

carrier-neutral data centers, 436–437

carrier-specific data centers, 436–437

case diagrams, 76

case sensitivity

PHP variable and member names, 162
 templates, 367

case study

building application. See building application deployment. See deployment project management. See project management project overview. See project overview project planning. See project planning systems architecture. See systems architecture

categories

content management, 572
 sections vs., 578–579

categories, project bug, 550–554

Cc: (carbon copy) recipients, e-mail, 260–262, 267

CDNs (content delivery networks), 615

cellular phones. See SMS (Short Message Service) text messaging

CGI binary, installing PHP as, 673–674

change management, 428–429

CIR (committed information rate)

bandwidth and, 596
 calculating, 437

class diagrams, UML

builder pattern, 97
 composite pattern, 76, 82
 decorator pattern, 90
 designing event-driven solution, 191
 implementation of, 39–42
 modeling domain, 35–37
 observer pattern, 83
 relationships, 37–39
 using Collection class, 119–120

class hierarchy, communications

Communication class, 255–259
 EmailRecipient class, 253–255
 overview of, 251
 Recipient class, 252–253

class keyword, 7

class type hints, 86

`class.Dispatcher.php`, **192–193**

classes

adding methods to, 6
 adding properties to, 8–10
 benefits of OOP, 4
 constructing objects with, 6–7
 contact manager application, 50–51
 creating, 7–8

classes (continued)

- defined, 5
- MVC design pattern model, 306–307
- MVC in Web applications, 308
- MVC mini toolkit, defining constants for, 312–314
- overview of, 6
- PHP lacking predefined, 132–133
- types of, 155–157
- unit testing, 344–346

`class.Event_Handler.php`, **194**

`class.Handler_Edit.php`, **195**

`class.Handler_View.php`, **194–195**

cleaning up application

- code debt, 475–476
- refactoring code, 476–482

client

- bugs reported by, 429–430
- disputes from differences in interpretation, 429
- gathering requirements from, 32
- project team role of, 414
- specification changes after sign-off, 429

client-side developers, choosing, 412

`close()` function, **291**

clustering, 600, 610–611

CMS (content management system)

- definition of, 571–573
- Drupal, 575–577
- ExpressionEngine, 580–583
- Frog CMS, 583–585
- history of, 573–575
- Joomla!, 578–580
- overview of, 575

CMS (content management system), building own

- administrators and privileges, 588–589
- content-related issues, 586–587
- overview of, 585
- templates, 590–591
- workflow, 589–590

CNET, history of, 574

code

- collapsed, 655
- debt, 475
- extreme programming, 427
- fixing holdups in, 665–666
- frameworks separating display logic and, 356–357
- reuse, 4, 16, 173
- Ulysses framework, 376
- unit testing picking up errors in, 540

- code completion, Zend, 649**
- code duplication, ORM limitations, 172**
- code editors**
 - CMS, 588
 - Komodo, 655–656
 - Zend IDE, 649–650
- code folding, Komodo, 655**
- collapsed code, 655**
- Collection class. See collections**
- collections**
 - Collection class basics, 106–109
 - Collection class design, 105
 - Collection class improvements, 125–126
 - Collection class purposes, 103–104
 - implementing lazy instantiation. See lazy instantiation
 - storing recipients using, 256–257
 - summary, 126
 - using Collection class, 108–109, 119–125
- co-located servers, hosting PHP applications, 595**
- comments, version control, 646**
- commercial terms, receiving brief of, 403–404**
- commit() method, PDO, 142–143**
- committals, PDO, 142–143**
- committed information rate (CIR)**
 - bandwidth and, 596
 - calculating, 437
- commodity hardware, hosting, 597**
- Communication class**
 - EmailCommunication class, 259
 - getting e-mail message across, 267
 - overview of, 255–256, 257–259
 - SMS text messaging, 270
 - storing recipients using collections, 256–257
 - TemplatedEmailCommunication class, 268–270
- community skills, PHP professionals, 630–631**
- Component class, composite pattern, 77**
- component diagrams, UML, 47–48**
- component failure, 602**
- components, spiral process, 417–418**
- Composite objects, composite pattern, 77**
- composite pattern, 76–83**
 - considerations, 82–83
 - implementation of, 78–82
 - overview of, 76–77
- composites, UML class diagrams, 38–39, 53–55**
- concrete constructor, LoggerBackend class, 213**

concurrent versioning

- choosing, 640–641
- conflicts, 638–640
- example of, 636–638
- exclusive vs., 636

Concurrent Versioning System (CVS)

- version control software, 644
- Zend integration with, 654

conditionals, Smarty, 337–338**connection strings**

- parsing, 210
- redesigning `Logger` class to use, 210–213

connectivity

- PHP database, 128–129
- physical environment and, 436–439
- systems architecture, 432
- testing PDO, 136
- using persistent connections, 147–148

Console pane, Zend, 653**constants**

- MVC mini toolkit, 312–314
- PDO, 143

Constraint class, 323–324**ConstraintFailure class, 323–324****`_construct()` function, 11–12, 29****constructors**

- creating contact manager, 57
- `DataBoundObject` class, 181–182
- emailing users, 261–262
- `HTTPSession` class, 301
- implementing business objects, 170–171
- inheritance using, 19–20
- instantiating objects with, 11–12

contact manager, creating

- contact type classes, 58–62
- `DataManager` class, 62–63
- description of, 49–50
- `Entity`, `Individual` and `Organization` classes, 64–71
- `PropertyObject` class, 56–58
- UML diagrams for, 50–56

content

- in content management, 572
- maximizing infrastructure power with, 614–615
- types of, 586–587

content delivery networks (CDNs), 615**content management. See CMS (content management system)****contributor, CMS building, 588****controller, MVC design pattern**

- file naming conventions, 310
- introducing, 306
- overview of, 307–308
- `search.php`, MVC mini toolkit, 324–326
- `searchresults.php`, MVC mini toolkit, 327–329
- Web applications, 308

controls

- adding to page in `Ulysses`, 367–371
- Prado Framework, 378

cookies

- defined, 280
- discovery of valid sessions, 283–284
- session perpetuation using, 280–281
- using supplemental keys, 283

CREATE DATABASE message, 128–129**Create New Project, Mantis, 550****creative specification design plan, 422****cross-platform support, PHP, 622–623****CRUD acronym, business objects, 161–162, 166–168****Customer Contact Report application**

- cleaning up application, 475–483
- finishing iteration, 483–492
- release planning, 450–451
- starting project. See starting project, building application
- starting the work. See starting work, building application

CVS (Concurrent Versioning System)

- version control software, 644
- version control using, 562–563
- Zend integration with, 654

D**data binding, business objects, 160–162****data centers, 436****data sources, limitations of PDO, 151****database**

- choosing type for PHP installation, 674–675
- clustering and replication, 610–611
- connecting PHP session management to, 290–292
- connectivity, 128–132
- creating for data binding, 160–161
- database support, 128
- development, 559–560
- fixing holdups in, 664–665

database (*continued*)

- Flickr configuration, 616
- functions specific to, 147
- high-performance architecture tips, 668
- limitations of ORM, 172
- logging to tables, 215–219
- maintaining job queue to hold e-mails, 271–272
- PDO specific functions, 147
- performance bottlenecks in queries, 663–664
 - for PHP application platforms, 599–600
- PostgreSQL, 128–129
- problem with PHP and, 132–133
- servers, 434
- session performance, 303
- using wrapper classes, 133–134

database abstraction

- framework for, 359
- overview of, 134–136
- in Ulysses framework, 377
- using ORM for. *See* ORM (Object Relation Mapping)

database source name (DSN), 137, 148

DataBoundObject **class**

- benefits, 184–185
- code, 175–178
- design, 174
- example implementation, 178–180
- framework for, 358
- overview of, 173
- understanding how it works, 180–184

DataManager **class**

- creating contact manager, 62–63
- indicating in class diagrams, UML, 52–55
- updating UML diagrams, 64

DataSource **object**

- decorator pattern, 92
- observer pattern, 87–89

date_default_timezone_set() **function**,

Logger **class**, 207

dayCalc() **function**, 524–526

db_changes **folder**, 560

db_connect() **function**, 194

DEBUG **function**, 220–221

Debug Output pane, Zend, 653

Debug window, Zend, 651

Debugger **class**, 219–222

debugging. *See also* logging; testing

- creating mechanism for, 219–222
- with Komodo, 656
- MVC design pattern and, 306

- testing phase plan, 424–425

- with ZDE, 651–654

debugPrint() **function**, 220–222

decision points, UML activity diagrams, 44

Decorator **class**, 90, 93–95

decorator pattern

- considerations, 94–95
- implementation of, 92
- overview of, 89–91
- using, 93–94

dedicated data centers, 436

dedicated load balancing, 435–436

definitions **element**, WSDL, 233–234

delivery time, 660

dependencies, UML component/deployment diagrams, 48

deployment

- application frameworks, 379–380
- devising development environments, 555–560
- overview of, 555
- source code workflow. *See* source code workflow, deployment
- summary, 567

deployment diagrams, UML, 47–48

design

- case study development approach, 388
- extreme programming, 426
- implementing business objects, 157

design patterns

- builder pattern, 96–100
- composite pattern, 76–83
- decorator pattern, 89–95
- facade pattern, 95–96
- MVC. *See* MVC (model-view-controller) design pattern
- observer pattern, 83–89
- overview of, 75
- reactor pattern, 199
- summary, 100–101

Design Patterns: Elements of Reusable Object-Oriented Software (Gamma, Helm Johnson and Vlissides), 630

design process, 422–423

destroy() **function**, 291

destroying objects, 12–15

_destruct() **function**, 12, 14, 29

destructor, business objects, 171–172

developer, career as. *See* professional software developer

development database, 559–560

development environments

- development databases, 559–560
- live production environment, 559
- live staging environment, 557–558
- overview of, 555
- simplicity in PHP, 619–620
- studio development environment, 556
- studio staging environment, 557

development servers, version control, 642**dictionary attacks, 282****digital signatures, 273****Director class, 97–98****disk fault tolerance, mitigation with, 606–607****Dispatcher class**

- `class.Dispatcher.php`, 193
- designing event-driven solution, 189–191
- event-driven programming considerations, 198–199

display logic

- frameworks separating code and, 356–357
- Ulysses framework, 376

`$displayHash`, 325, 326, 329**`displaySmartyPage()` function, 481–482****DKIM (Domain Keys Identified Mail), 273–274****DLLs, 680****DNS load balancing, 435, 603–604****domain experts, project management, 395–396****Domain Keys Identified Mail (DKIM), 273–274****domains, 157**

- defined, 32
- modeling in class diagrams, 35–37

`draw()` method, decorator pattern, 89–90, 91–92**drop-downs, forcing submits with, 483****Drupal**

- creating content with, 576
- managing content with, 577
- sample layout, 575

DSN (database source name), 137, 148**dual Ethernet segmentation, 609****dwell time, calculating CIR, 437****E****`_e` ssh directive, `rsync`, 565****eCircle, 274****editing code**

- Komodo, 655–656
- Zend IDE, 649–650

ellipses (. . .), 37**e-mail**

- broadcast providers, 274
- notification through, 248–249
- recipients, 251
- worms, 284

EmailCommunication class

- building test version, 259–263
- getting message across, 263–268
- overview of, 259
- using MIME, 270
- using templates, 268–270

emailing users

- blocking activity, 271–272
- building test version, 259–263
- deliverability, 272–274
- getting message across, 263–268
- overview of, 259
- use of templates, 268–270
- using MIME, 270

EmailRecipient class, 253–255**EmailRecipientCommunication class, 259****encapsulation**

- changes in PHP6, 29
- defined, 6
- overview of, 27–28

Entity class

- creating contact manager, 52–55, 59
- developing, 65–71

environments, development. See development environments**`errorCode()` method, PDO, 143–145****errors, handling PDO, 143–145****errors log, 209****escape, 141****Ethernet segments, 608–610****event-driven programming**

- designing, 189–191
- implementing, 192–195
- overview of, 187
- pausing for thought, 198–199
- security, 196–198
- summary, 199
- understanding events, 187–188

events, 187–188**exception handling, SOAP client, 239–240****exceptional notifications, 248****exclusive versioning, 636, 640–641****`exists()` method, 108****expectations, setting during pitch, 409****expiration times, sessions, 285**

exponential sequence, 312

ExpressionEngine

- adding content in, 581–582
- template functions, 582–583
- Web site features, 580–581

extends **keyword, inheritance, 16, 17**

extension, PHP as, 128

extensions **directory, 680**

exterior networks, PHP, 595–596

external data sources, performance tuning, 666

extreme programming. See XP (extreme programming)

Extreme Programming Explained: Embrace Change, Second Edition (Addison-Wesley, 2004), 387, 426

F

facade pattern, 95–96

Facebook, latency and, 596

false start, project history, 396

fault tracking, QA, 543–544

faultcode property, SoapFault class, 239–240

faultstring property, SoapFault class, 239–240

fax

- e-mail-to-fax gateways, 271
- recipients, 251

fclose() method, Logger class, 202

feature creep, 399

feature sets, and PDO, 150–151

fetch() method, PDOStatement class, 139

fetchall() method, PDOStatement class, 139

file logging

- example file system layout, 202–203
- simple, 201–202
- using Logger class, 203–208

File Transfer Protocol (FTP), warning, 567

files, splitting in MVC, 309

finishing iteration, building application, 483–492

firewall, hardware requirements, 436

Flickr, 616

fopen() method, Logger class, 202

forks, UML activity diagrams, 44

form persistence, frameworks, 358, 376

framework usability, unit testing, 351

frameworks. See application frameworks

Frog CMS

content management simplification, 583

managing pages, 584

snippets, 584–585

front controller, Zend Framework, 379

FTP (File Transfer Protocol), warning, 567

functional compliance quality, 538, 541–542

functional specification, 420–421

functional test form, 541

functional testing

considerations, 542

functional test form, 541–542

overview of, 541

plan, 424

unit testing reducing burden of, 352

usability testing vs., 543

future plans, formal brief of, 404

fwrite() method, Logger class, 202

G

GANTT chart, 409

gc() function, 292

Gd (graphics library), 666

generalization, UML class diagrams, 38, 52–54

generateSqlDelete() function, 487

generateSqlUpdate() function, 487

geographic load balancing, 605–606, 617

GET requests, HTTP

how it works, 276–278

performance bottlenecks, 659–661

as stateless, 275

Ulysses framework, 366

GetAccessor() method, 160, 183

getConnection() function, 62–63

getDescription() method, 81–82

_getFunctions method, SoapClient, 228, 229, 230

getInstance() method, Logger class, 205–206, 210–213

getItem() method, Collection class, 105–109

_getLastRequest method, SoapClient, 228

_getLastResponse method, SoapClient, 228

getSessionIdentifier() method, HTTPSsession class, 302

getSqlWhere() method, 487–489

getStartDateOffset() method, 473

getStringRepresentation() method, 252, 254–255

_getTypes method, SoapClient, 228

`getUserID()` **function**, `HTTPSession` **class**, **302**
`getUserObject()` **function**, `HTTPSession` **class**, **302**
GNU/Linux, **672, 675**
good session practice, **199**
grace during failure, QA, **538**
graceful degradation, **143**
graphics library (Gd), **666**

H

hackers, **281–285**
`_handle` **method**, `SoapServer`, **229**
`handle_the_event()` **function**, **193–194**
`handled_event()` **method**
 `class.Dispatcher.php`, 193
 `class.Event_Handler.php`, 194
 `class.Handler_Edit.php`, 195
 `class.Handler_View.php`, 194–195
 `interface.Handled.php`, 193
handlers
 `class.Dispatcher.php`, 193
 `class.Event_Handler.php`, 194
 `class.Handler_Edit.php`, 195
 `class.Handler_View.php`, 194–195
 designing event-driven solution, 189–191
 event-driven programming considerations, 198–199
 `interface.Handled.php`, 193
handoff, defined, **596–597**
handover process, **425**
hardware
 determining requirements, 434–436
 high-performance tips, 667
 load balancing as mitigation technique, 604–605
 performance problems, 662–663
 for PHP application platforms, 597
hashes, Smarty, **337**
heartbeat link, **604**
high bandwidth, **595**
high traffic/high availability
 Flickr, 616
 high availability, 594
 high traffic, 594–595
 impact factors. See impact factors
 mitigation techniques. See mitigation techniques
 overview of, 593, 615

platforms. See platforms
 real-world examples, 615–618
 summary, 618
 understanding the terms, 593–595
 Wikipedia, 616–618
high-quality product, QA, **536**
history of project, **396–397**
hosting
 business requirements, 433
 systems architecture, 432
HOSTS file, **283–284**
HTML, `search.phtml`, **326–327**
HTTP latency, **601**
HTTP requests
 blocking activity, 271
 how it works, 276–278
 overview of, 275
 sessions and, 277–278
HTTPSession class
 database schema, 292–293
 how it works, 300–302
 `HTTPSession.phpm` code, 293–297
 overview of, 292
 performance considerations, 302–303
 summary, 304
 testing, 297–300
`HTTPSession.phpm` **code**, **297–300**

I

`$ID` **member variable**, **170**
IDEs (integrated development environments)
 choosing, 647–648
 Komodo, 654–657
 other types of editors and, 657
 summary, 658
 Zend Studio, 648–654
IIS (Internet Information Services), hosting on Windows with, **673**
impact factors, **601–602**
`impress()` **function**, `HTTPSession` **class**, **301–302**
include, use case diagrams, **34–35**
`index.php`, **478**
Individual class, **51–53, 68–71**
information architects, choosing, **412**
information architecture design plan, **422–423**
infrastructure, business requirements, **433**

inheritance

- class diagrams for, 38, 52–55
 - defined, 5
 - overriding methods, 20–23
 - overview of, 15–20
 - preserving parent's functionality, 23–24
 - understanding, 24–25
- initialization, of objects, 11–12**
- inline method, refactoring, 482**
- INNER joins, database holdups, 665**
- input, processing, and output (IPO) model, 308**
- INSERT statements, database connectivity, 131**
- installation, Mantis, 545–547**
- installation, PHP**
- choosing database, 674–675
 - choosing Web server, 673–674
 - downloading and installing PostgreSQL, 675–677
 - installing PHP and Apache, 677–679
 - installing support libraries, 677
 - steps for, 671–672
 - summary, 680
 - testing installation, 679
 - with UNIX servers, 672–673
 - when Windows is needed, 679–680
- installation, PHPUnit, 347**
- installation, Smarty, 334–335**
- instantiation**
- lazy. See lazy instantiation
 - of objects, 6–7
 - singleton, 149
- Instrument interface**
- implementation of, 39–41
 - indicating associations, 37–38
 - indicating composites, 38–39
 - using composite pattern. See composite pattern
- integrated development environments. See IDEs (integrated development environments)**
- interface.Handled.php, 193**
- interfaces**
- class diagrams indicating, 38, 77
 - component/deployment diagrams indicating, 48
 - creating Recipient interface, 252–253
 - defined, 6
 - designing event-driven solution, 191
 - designing for classes, 344–345
 - implementing event-driven solution, 193
 - overview of, 25–27
- interior network, application platforms, 596–597**
- InterLeaf, 573**
- internal member variables**
- defined, 10

- encapsulation and, 28
 - overview of, 25–27
- Internet Information Services (IIS), hosting on Windows with, 673**
- interviewing client, requirements gathering, 32**
- introduction, pitch, 409**
- investment summary, pitch, 409**
- IP addresses, good session practice, 286**
- IPO (input, processing, and output) model, 308**
- is_callable() function, 115**
- isEmpty() function, 496**
- isLoggedIn() function, 302**
- isset() method, 172**
- isValid() method, 252, 255**
- iterations**
- building. See building application
 - release planning, 450–451

J

- Java, support for, 624**
- joins**
- OUTER joins, and database holdups, 665
 - UML activity diagrams, 44
- Joomla!**
- control panel, 580
 - creating content with, 579
 - Joomla!-powered site, 578
- journaling of files, 635–636**
- junk mail**
- broadcast e-mail providers and, 274
 - Domain Keys Identified Mail and, 273–274
 - limiting deliverability of, 271–272
 - Sender ID and, 273
 - SPF records and, 273

K

- KeyInUseException class, 107–109**
- keys() function, 108**
- Komodo**
- editing code, 655–656
 - managing projects, 654–655
 - overview of, 654

L

- languages, weakly typed in PHP, 621**
- large objects, PDO, 146–147**
- lastInsertID() method, business objects, 171**

latency (ping time), bandwidth and, 596**lazy instantiation**

- callbacks, 110–115
- defined, 104
- implementing, 109–110
- implementing business objects, 173
- setLoadCallback method of `Collection` class, 115–119

lead architect

- choosing, 411–412
- managing build process, 424
- producing ten point plan, 423

libicu, support libraries, 677**libicu-devel, support libraries, 677****libraries**

- external, PHP not requiring in Windows, 680
- implementing SOAP in PHP 6, 227
- installing support, 677

lifeline, UML sequence diagrams, 45**lighttpd Web server, 599****linear arrays, Smarty, 336****linguistic simplicity, PHP, 621–622****link aggregation (NIC teaming), 606****Linux Virtual Server (LVS), 617****Listener interface, observer pattern, 83–86, 88–89****live production environment, 559****live staging environment, 557–558****Load() method, DataBoundObject class, 181–182****load balancer, 604****load balancing**

- dedicated, 435–436
- DNS-based, 435, 603–604
- hardware, 604–605
- overview of, 603
- software-based, 435

load testing

- high-performance architecture using, 668–669
- for quality assurance, 542
- testing phase plan, 424–425

load tolerance, QA, 539**LoadRunner, 425****LOBs (large objects), 146–147****lodging_sun_0, 500–501****log out, 478–479****Logger class**

- extending, 208–209
- LoggerBackend class, 213
- logging to database table, 215–219
- overview of, 203–208

parsing connection string, 210

redesigning to use connection string, 210–213

subclassing `LoggerBackend` class, 213–215

LoggerBackend class

logging to database table, 215–219

overview of, 213

redesigning to use connection string, 212–213

subclassing, 213–215

logging. See also debugging

to database table, 215–219

example file system layout, 202–203

`Logger` class, 203–208

`LoggerBackend` class, 213

overview of, 208–209

parsing connection string, 210

redesigning `Logger` class to use connection string, 210–213

simple file, 201–202

subclassing `LoggerBackend` class, 213–215

login

cleaning up application, 477–479

configuring Mantis, 547–548

creating screen, 460–464

database connectivity in PHP, 130

good session practice, 286

in usability testing, 543

logMessage() method

`Logger` class, 202–203, 206–208

`LoggerBackend` class, 213

look and feel, formal brief of, 404–405**LVS (Linux Virtual Server), 617****M****mail() function, 250****maintenance management, 433, 439****manageBackends() function, Logger class, 213****managed servers, hosting PHP applications, 595****managing projects. See project management****Mantis, 544–554**

annotating bugs, 553

assigning bugs, 553

creating and editing users, 548–549

creating new projects, 550

getting the most out of, 553–554

logging in as administrator, 547–548

overview of, 544–547

reporting bugs, 552–553

Mantis (continued)

- resolving bugs, 553–554
- setting up project bug categories, 550–552
- `MarkForDeletion()` **method, business objects, 171**
- MDB2, PEAR repository, 133–134**
- MediaWiki, 617**
- member variables**
 - implementing business objects, 170
 - internal, 10
 - protecting access to, 10–11
 - `Request` class, 312–314
- `memcached` **server, 303, 617–618**
- memory**
 - performance tuning and, 666
 - session performance and, 303
- message **elements, WSDL, 235–236**
- messages, e-mail, 250, 263–268**
- methods**
 - accessor. See accessor methods
 - adding, 8
 - adding to classes, 6
 - `Collection` class, 106–108
 - overriding, 20–23
 - `request` object, 320–322
- Microsoft Project, 428**
- Microsoft Visual SourceSafe, 643–644**
- Microsoft Windows**
 - operating systems for, 598
 - transferring PHP from UNIX to, 679–680
 - using UNIX Web servers vs., 672–673
- milestones, establishing, 400–401**
- MIME, for HTML e-mail, 270**
- mitigation techniques**
 - asset servers, 615
 - caching, 613–614
 - content delivery networks, 615
 - content generation, 614–615
 - database clustering and replication, 610–611
 - disk fault tolerance, 606–607
 - geographic balancing, 605–606
 - load balancing, 603–605
 - multi-segment topologies, 608–610
 - NIC teaming, 606
 - overview of, 603
 - power redundancy, 607–608
 - traffic calculus, 611–613
- mock objects, building applications, 529–533**
- `mod_rewrite` **support, Apache, 362**

- model, MVC design pattern, 306–308, 310**
- modeling domain, software design for, 35–37**
- model-view-controller. See MVC (model-view-controller) design pattern**
- modularity, OOP, 4–5**
- monthly transfer, calculating, 438**
- MS SQL Server, 128**
- multidimensional arrays, Smarty, 340–341**
- multi-segment topologies, Ethernet, 608–610**
- multitasking, illusion of, 601–602**
- MVC (model-view-controller) design pattern**
 - controllers, 307–308
 - model, 306–307
 - overview of, 306
 - in PHP, 309–311
 - view, 307
 - in Web applications, 308
 - Zend Framework as traditional, 378–379
- MVC (model-view-controller) design pattern, mini toolkit**
 - constants, 312–314
 - `Constraint` class, 322–324
 - overview of, 311–312
 - PRG, 331–332
 - `Request` class, 314–322
 - `search.php`, 324–326
 - `search.phtml`, 326–327
 - `searchresults.php`, 327–329
 - `searchresults.phtml`, 330
 - trying it, 330–331
- MyProject window, Zend, 649**
- MySQL**
 - calling stored procedure in, 148
 - database clustering and replication, 611
 - Flickr use of, 616
 - `php.net` reference site for, 128

N

- naming conventions**
 - cookies, 281
 - database fields, 57
 - determining unique values of individual cells, 500–501
 - MVC design pattern for files, 309–310
 - PHP variable and member names, 162
- native templating**
 - overview of, 310
 - pitfalls of, 333–334
 - recapping, 333

- true templating vs., 334
 - when to use Smarty vs., 342
 - navigability, association lines and, 37–38**
 - Net_Smtp package, 263–268**
 - networks**
 - high traffic/high availability and, 602
 - physical environment and connectivity, 437
 - planning systems architecture, 433
 - NIC teaming (link aggregation), 606**
 - nix systems, 672**
 - nodes**
 - in Drupal, 576–577
 - UML component/deployment diagrams, 47–48
 - notification, 248–249**
 - `notifyObservers()` **method**, *Observable* **interface, 88**
 - NuSOAP, 227**
- O**
- object composition, 75**
 - object diagrams, 81**
 - object overloading, 160**
 - Object Relation Mapping.** See **ORM (Object Relation Mapping)**
 - objectives, setting during pitch, 409**
 - object-oriented programming.** See **OOP (object-oriented programming)**
 - objects**
 - adding methods to, 8
 - adding properties to, 8–10
 - creating classes, 7–8
 - creating contact manager. See *contact manager*, **creating**
 - defined, 5
 - destroying, 12–15
 - initializing, 11–12
 - overview of, 6–7
 - protecting access to member variables, 10–11
 - system, 71–73
 - `$objPDO` **member variable, 170**
 - Observable* **interface, observer pattern, 83, 87–89**
 - Observer* **interface, observer pattern, 83–86, 88–89**
 - observer pattern**
 - connecting *Observer* and *Observable*, 88–89
 - considerations, 89
 - DataSource*, 87–88
 - overview of, 83
 - widgets, 84–86
 - Once and Only Once refactoring, 479–481**
 - online references**
 - BaseCamp, 428
 - broadcast e-mail providers, 274
 - Domain Keys Identified Mail, 274
 - eFax, 271
 - enabling PHP support, 136
 - extreme programming, 428
 - libraries implementing SOAP in PHP 6, 227
 - LoadRunner, 425
 - Mantis, 424, 544
 - memcached server, 303
 - MIME, 270
 - PDO documentation, 151
 - PHP supported drivers, 137
 - php.net reference site, 128
 - PMI (Project Management Institute), 416
 - PostgreSQL source code, 675
 - Prado, 377–378
 - Sender ID, 273
 - Smarty, 334, 342
 - SoapClient* and *SoapServer* functions, 227
 - SPF records, 273
 - SQL injection attacks, 141
 - Ulysses, 361
 - WSDL, 233
 - Zend Framework, 379, 648
 - on-site customers**
 - case study development approach, 388
 - XP, 426
 - on-site exploitation, 284**
 - OOP (object-oriented programming)**
 - classes, 6
 - encapsulation, 27–28
 - inheritance. See **inheritance**
 - interfaces, 25–27
 - objects. See **objects**
 - overview of, 5–6
 - PHP 6 changes to, 29–30
 - summary, 30
 - understanding, 3–5
 - opcode caching, 662**
 - `open()` **function, 291**
 - operating system, choosing for PHP installation, 672–673**
 - operating systems, PHP application platforms, 598**
 - operations, class diagrams, 36**
 - Oracle, 128**

Organization class, 52–53, 68–71

ORM (Object Relation Mapping)

business objects and. See business objects

DataBoundObject class. See

DataBoundObject class

as framework, 358–359

overview of, 155

summary, 185

types of classes, 155–157

in Ulysses framework, 377

OUTER joins, and database holdups, 665

Outline window, Zend, 651

output, capturing Smarty, 341

overhead, and PHP, 619–620

overriding methods, of parent class, 20–23

overtime, XP and, 427

P

page map design plan, 422–423

pages, Prado Framework, 378

parent class

overriding methods of, 20–23

preserving functionality of, 23–24

`parent::[function name]`, 23

parse() method, email output, 269

parse request test, satisfying, 503–505

parse_url() function, 210, 212

parsing, connection strings, 210

passwordless SSH, 566–567

passwords

cleaning up application, 477

good session practice, 286

how HTTP works, 277–278

Mantis login as administrator, 547–548

validation of, 249

patches, performance

fixing code holdups, 665–667

fixing database holdups, 664–665

testing, 667

paths, transferring PHP from UNIX to Windows, 679–680

Pattern-Oriented Software Architecture, Volume 2, Patterns for Concurrent and Networked Objects (Wiley, 2000), 199

PDFs, creating pitch documents as, 410

PDO (PHP Data Objects)

constants, 143

database abstraction, 134–136

database-specific functions, 147

defined, 135

enabling PHP support, 135–136

executing queries, 138

as framework for database abstraction, 359

handling errors, 143–145

large objects, 146–147

limitations of, 150–151

overview of, 127

PDO class, 137

PDOStatement class, 139

persistent connections, 147–148

PHP and databases, 128–134

prepared statements, 139–141

session performance and, 302–303

singleton instantiation, 149

stored procedures, 148–149

summary, 152

transactions and committals, 142–143

write-only statements, 141–142

PDO class

constants, 143

database-specific functions, 147

executing queries, 138

handling errors, 143–144

overview of, 137

PDOStatement class, 139

prepared statements, 139–141

transactions and committals, 142–143

write-only statements, 141–142

PDOFactory, Ulysses framework, 375

PDOStatement class

database-specific functions, 147

handling errors, 143–144

overview of, 139

querying database, 138

PEAR, handling MIME with, 270

PEAR repository, 680

PEAR::SOAP, 227

perfect-engineering time unit, 444

performance

HTTPSession class and, 302–303

limitations of basic PHP sessions, 290

prepared statements and, 141

performance tuning PHP

anatomy of PHP requests, 661–663

patching, 664–667

preemptive avoidance and, 667–669

tracking down bottlenecks, 663–664

types of performance bottlenecks, 659–661

permissions, Smarty installation, 335

persistence, Ulysses framework, 372–373

persistent connections, PDO, 147–148

PHP

content management. See CMS (content management system)

cross-platform support, 622–623

linguistic simplicity, 620–622

low overhead, 619–620

MVC in, 309–311

overview of, 619

power, 624–625

summary, 625

ubiquity, 624

Zend Functions, 651

.php (controllers)

MVC rules, 310

search.php, 324–325

searchresults.php, 327–329

PHP, professional career in

motivation, 627–628

overview of, 627

software developer, 628–630

PHP Data Objects. See PDO (PHP Data Objects)

.phpm (classes), MVC rules, 310

PHPUnit

customer contact tests, 465–467

installing, 347

introduction to, 346–347

test-driven development using, 425

unit testing, 346–350

writing tests, building application, 453–460

.phtml (templates)

MVC rules, 310–311

search.phtml, 325–327

searchresults.phtml, 330

physical environment, and connectivity, 436–439

physical intrusion, and session security, 283–284

pitch

functional specification using, 420

vs. quotes, 406

vs. specifications, 406–407

structuring, 409–410

when to go extra mile, 408

when to say no, 408–409

who to involve, 407

platforms

application servers, 599

databases, 599–600

exterior network, 595–596

hardware, 597

interior network, 596–597

operating systems, 598

overview of, 595

software architecture, 600

Web server, 598–599

PMI (Project Management Institute), 415

polymorphism, 5

portType element, WSDL, 236

POST requests, HTTP

how it works, 276–278

as stateless, 275

types of performance bottlenecks, 659–661

Ulysses framework, 366

PostgreSQL database

calling stored procedure in, 148

configuring Mantis for first time, 545–546

connecting and disconnecting via PDO, 136

connecting PHP session management to, 290–292

connectivity in PHP, 128–132

database clustering and replication, 611

destroying objects, 13–15

downloading and installing, 675–677

HTTPSession class, 292

overview of, 12–13

PHP and, 128–129

php.net reference site, 128

reasons to choose, 674–675

schema, 292–293

post-redirect-get (PRG) design pattern, 331–332

power, of PHP, 624–625

power redundancy, 607–608

Prado Framework, 377–378

Preferences window, Zend, 649

prepare() method, PDO, 140

prepare() method, PDOStatement class, 139

prepared statements, PDO, 139–141

prerequisites, project management, 397

PRG (post-redirect-get) design pattern, 331–332

primary keys, fixing database holdups, 664

primary recipients, e-mail, 260–262, 267

printArray() method, debugging, 221–222

privacy issues, cookies, 281

private member variables

encapsulation and, 28

HTTPSession class, 300

overview of, 11

privileges, building CMS, 588–589

process

- build phase, 424
- choosing in project planning, 415–416
- design phase, 422–423
- handover, 425
- making decision on, 419
- overview of, 419
- specification phase, 419–421
- spiral, 417–418
- testing phase, 424–425
- waterfall, 416–417

processing time, scripts, 660

production environment, 555

production server, 555

professional software developer

- academic skills, 630
- community skills, 630–631
- more than web development skills, 628–629
- overview of, 628
- soft skills, 629
- summary, 631

profiler, Zend, 654

programming methodology, 424–426

project management

- anticipated project prerequisites, 397
- choosing team, 410–414
- in CMS building, 588
- constructing pitch, 406–410
- history of project, 396
- Komodo, 654–655
- overview of, 393–394
- who project is for, 394–396
- why project is happening, 394
- Zend IDE, 649

Project Management Institute (PMI), 415

project management, receiving formal brief

- budget, 401–403
- business requirements, 398–399
- commercial terms, 403–404
- future plans, 404
- look and feel, 404–405
- overview of, 397–398
- scope, 399–400
- summary, 405–406
- support, 405
- technology, 405
- timelines, 400–401

project manager, 410–411

project overview

- development approach, 387–391

- overview of, 383–384

- summary, 392

- Widget World, background of, 384–386

- Widget World landscape, 386–387

project planning

- build phase, 424

- build process methodologies, 425–428

- change management, 428–430

- choosing process, 415–416

- design phase, 422–423

- handover, 425

- making decision, 419

- overview of, 419

- specification phase, 419–421

- spiral process, 417–418

- summary, 428–430

- testing phase, 424–425

- waterfall process, 416–417

Project tab, Zend, 651

properties

- adding to classes, 8–10

- of classes, 6

- monitoring, 173

- using accessor methods for, 11

PropertyObject class

- contact type classes, 58–62

- creating contact manager, 56–58

- updating UML diagrams, 64

protected member variables

- creating contact manager, 57

- DataBoundObject class, 181

- encapsulation and, 28

- overview of, 11

prototyping, business objects, 158–160

pserver, 643

public members

- encapsulation and, 28

- member variables or methods, 11

punctuation, reduced in PHP, 621–622

Python, support for, 624

Q

QA (quality assurance)

- introduction to, 535–536

- measurable and quantifiable quality, 538–540

- quality defined, 537

- testing. *See* testing

- with unit testing frameworks, 351–352

- why you should aim high, 536–537

queries

- executing using PDO objects, 138
- fixing holdups, 664–665
- high-performance architecture tips, 668
- limitations of ORM, 172
- performance bottlenecks in, 663–664
- syntax limitations of PDO, 150

quotes, vs. pitches, 406**R****_r flag, rsync, 565****rack space, 438****RAID (Redundant Array of Independent Disks), 607****reactor, 189****reactor pattern, 199****read() function, 291****realizations, class diagrams, 38****real-world tolerance (RWT), QA, 538****recalculate() function, 523, 525****Recipient class, 252–253****Recipient interface, 252–255****recipients. See also emailing users**

- comparison of, 251
- defined, 250
- EmailRecipient class, 253–255
- Recipient class, 252–253
- storing using collections, 256–257

redraw() function, observer pattern, 89**redundancy, 433****Redundant Array of Independent Disks (RAID), 607****re-estimating, starting work, 474–475, 594****Refactoring (Fowler), 481–482****refactoring code**

- case study development approach, 388
- in extreme programming, 426
- finishing iteration, 483–492
- overview of, 476–482
- Travel Expense Report application, 509–514

Refactoring: Improving the Design of Existing Code (Fowler, Beck, Brant, Opdyke and Roberts), 630**references, book. See also online references**

- design patterns, 199
- extreme programming, 387, 426
- refactoring code, 481–482
- understanding XML, 226

register() method, Logger class, 210–213**regression testing, 351, 418****relation mapping, 162****relationship, UML class diagrams, 37–39, 52–55****releases, 387, 450–451****remote debugging, Zend, 654****removeItem() method, Collection class, 105–109****replication**

- databases and, 600
- defined, 611

Report Bug, Mantis, 552–553**Representational State Transfer. See REST (Representational State Transfer)****Request class, MVC, 314–322****Request element, WSDL, 234–235****requests, PHP**

- anatomy of, 661–663
- SOAP extension, 237–239

requirements

- implementing business objects, 166–169
- UML gathering phase, 31–33

resilience, systems architecture, 433**responses, PHP6 SOAP extension, 237–239****REST (Representational State Transfer)**

- implementing client, 244–245
- making server, 245–246
- overview of, 243
- PHP 6 and, 244

reusability, 4, 16, 173**roles, project**

- client, 414
- identifying, 395
- manipulating to advantage, 395–396
- team doubled up on, 413

rollback() method, PDO, 142–143**round-robin load balancing method, 436****rsync**

- basic usage, 564–565
- for keeping servers in sync, 566–567
- syntax examples, 565
- version control using, 564

Ruby, support for, 624**run-time configuration, Smarty, 340****RWT (real-world tolerance), QA, 538****S****SAPI module, installing PHP as, 673–674****Save() method**

- business objects, 171
- DataBoundObject class, 182

scheduled_email table, 272

scope

- of cookies, 283–284
- receiving formal brief of, 399–400

screen, creating, 399, 470–471

search.php, 324–326, 339–340

search.phtml, 326–327, 339–340

searchresults.php, 327–329

searchresults.phtml, 330

sections, categories vs., 578–579

secure_handler() method, 196–197

security

- business requirements, 434
- event-driven programming, 196–198
- planning systems architecture, 433
- session, 281–285, 290

SELECT query, ORM, 172, 173

SELECT statements, 171, 664

SELECT*, avoiding, 665

send() method, e-mail, 261, 263

Sender ID, 273

Sender Policy Framework (SPF) records, 273

senior designers, choosing, 413

sequence diagrams, UML, 44–46

sequential sequence, 312

server load, 601–602

\$_SERVER["HTTP_HOST"] property, 560

servers

- determining hardware requirements, 434
- physical location of, 436–439
- planning systems architecture, 432
- rack space considerations, 438

server-side includes (SSIs), 614–615

servlets

- session IDs, 378
- Ulysses framework, 366–372

session IDs

- overview of, 278
- session security and, 281–285
- storing in cookies, 280–281
- URL rewriting using, 279–280

session timeouts

- creating table to calculate, 292
- good session practice, 285
- overview of, 278

session_read_method(), 301

session_set_save_handler() method, 291, 292, 301

session_start() method, 288–289, 291

session-based load balancing method, 436

sessions

- basic, 288–290
- calculating CIR, 437
- calculating monthly transfer, 438
- definition of, 275, 278
- good practices, 285–287
- how HTTP works, 276–278
- overview of, 275
- perpetuating with cookies, 280–281
- perpetuating with URL rewriting, 279–280
- security, 281–285

sessions and authentication

- connecting PHP session management to database, 290–292
- creating Authentication class, 290
- database schema, 292–293
- good session practice, 287
- how HTTPSession class works, 300–302
- HTTPSession.php code, 293–297
- performance considerations, 302–303
- putting it all together, 304
- summary, 304
- testing HTTPSession class, 297–300
- using HTTPSession class, 292

_set() overload method, 302

SetAccessor() method, 160, 183

_setClass method, SoapServer, 229

setLoadCallback method, Collection class, 115–119

_setPersistence method, SoapServer, 229

shared network drives, 666

Short Message Service. See SMS (Short Message Service) text messaging

silent mode, PDO, 143

Simple Object Access Protocol. See SOAP (Simple Object Access Protocol)

simplicity, of PHP, 619–622

singleton pattern, 149, 205

sizeof() function, 108

small releases, XP, 426

Smarty

- associative arrays in, 337
- capturing output, 342
- conditionals in, 337–338
- finalized Travel Expense Report, 516–520
- functions, 341–342
- further reading, 342
- installing, 334–335
- linear arrays in, 336
- mocking up HTML screens with, 529–532

- multidimensional arrays, 340–341
- other templates, 342
- producing template-driven email output, 268–270
- rewriting `search.php` with, 338–339
- run-time configuration, 340
- `smartytest.php`, 335
- `smartytest.tpl`, 336
- traditional templating vs., 342
- true templating with, 334
- variable modifiers, 341
- `smartytest.php`, **335**
- `smartytest.tpl`, **336**
- SMS (Short Message Service) text messaging**
 - overview of, 270
 - recipients, 251
 - using as authentication, 250
- SMTP servers, 271**
- snippets**
 - CMS templates and, 590
 - Frog CMS management of, 584–585
- SOAP (Simple Object Access Protocol)**
 - making server, 240–243
 - overview of, 226
 - PHP 6 and, 226–227
 - PHP 6 extension for, 227–230
 - SMS text messaging interface via, 270
- SOAP (Simple Object Access Protocol), making client**
 - exception handling in, 239–240
 - overview of, 230–232
 - request and response envelopes, 237–239
 - WSDL document, 232–237
- `soap_functions.php`, **230**
- `SoapClient` **functions, 227, 228**
- `SoapFault` **object, 229, 239–240**
- `SoapHeader` **object, 229**
- `SoapParam` **object, 230**
- `SoapServer` **function, 227, 228–229**
- `SoapVar` **object, 229**
- sockets**
 - performance tuning, 666
 - response time, 601
- soft skills, of PHP professionals, 629**
- software. See also professional software developer**
 - architecture design plan, 423
 - choosing architects and engineers, 412
 - importance of architecture, 600
 - load balancing based on, 435
 - version control, 643–645
 - Wikipedia (MediaWiki), 617
- software developer. See professional software developer**
- source code repository, 477**
- source code workflow**
 - automated version control, 562–565
 - diagram, 561
 - overview of, 560–562
- SourceSafe, 643–644**
- spam**
 - broadcast e-mail providers and, 274
 - Domain Keys Identified Mail and, 273–274
 - limiting deliverability of, 271–272
 - Sender ID and, 273
 - SPF records and, 273
- specification phase, processes, 419–421**
- specifications**
 - bugs reported by client, 429
 - changes after sign-off, 429
 - disputes arising from differences in interpretation, 429
 - pitch vs., 406–407
 - revising, 428–429
- speed, and basic PHP sessions, 290**
- SPF (Sender Policy Framework) records, 273**
- spike**
 - starting project, 448–449
 - Travel Expense Report, 499–500
- spiral process**
 - choosing, 417–418
 - making decision on, 419
 - technical specification, 421
- spreadsheet, 520–526**
- SQL injection attacks, 141**
- Squid Web server, 599, 605**
- SSIs (server-side includes), 614–615**
- staging server, 642**
- starting points, UML activity diagrams, 42**
- starting project, building application**
 - collecting stories about new system, 442–444
 - estimation tips, 449–450
 - overview of, 441–442
 - refining estimations, 448
 - release planning, 450–451
 - the spike, 448–449
 - story weight estimation, 444–447
- starting work, building application**
 - creating login screen, 460–464
 - creating screen, 470–471

starting work, building application (continued)

- customer contact requirements, 464–465
 - customer contact tests, 465–467
 - feeding beast, 471–474
 - next story, 464
 - outlining details of a story, 451–452
 - PHPUnit, 453–460
 - re-estimating, 474–475
 - satisfying tests, 467–469
 - writing tests, 452–453
- state, 6**
- state diagrams, UML, 46–47**
- stateless protocol, HTTP as, 248, 275**
- static file server, Wikipedia, 617**
- static methods, 63, 113**
- storage requirements, 434, 437**
- stored procedures, PDO, 148–149**
- stories**
- estimating effort required to implement, 444–447
 - estimation tips, 449–450
 - outlining details of, 451–452
 - release planning, 450–451
 - spike, 448–449
 - starting project by documenting, 442–444
- `strftime()` **function, Logger class, 206–207**
- strings, Logger class, 207**
- `strip` **function, Smarty, 341**
- studio artists, 413**
- studio development environment, 556**
- studio staging environment, 557**
- StudleyCaps, 162**
- style guide design plan, 422**
- subclasses, LoggerBackend class, 213–215**
- submits, forcing, 483**
- subscriber, CMS, 588**
- subselects, fixing database holdups, 665**
- Subversion.** See **SVN (Subversion)**
- supplemental keys, session security, 283**
- support**
- after-production, 405
 - cross-platform, 622–623
- SVN (Subversion)**
- version control software, 644–645
 - version control using, 563
 - Zend integration with, 654
- swimlanes, UML activity diagrams, 44**
- syntax, commonly understood, 622**
- systems architecture**
- business requirements, 433–434

- definition of, 431
- hardware requirements, 434–436
- importance of, 432
- maintenance management, 439
- physical environment and connectivity, 436–439
- summary, 439

T

tables, database

- logging to, 215–219
- maintaining job queue to hold e-mails, 271–272
- session authentication and, 292–293

tagging

- in studio staging environments, 557
- version control technique, 645–646

taxonomy, organizing nodes, 577

team, project

- account manager, 411
- client-side developers, 412
- doubling up of roles, 413
- information architects, 412
- lead architect, 411–412
- project manager, 410–411
- senior designers, 413
- software architects and engineers, 412
- studio artists, 413
- working practices, 413

technical specification, 421–422

technology

- case study project overview, 391
- receiving formal brief of, 405

`TemplatedEmailCommunication` **class, 268–270**

templates. See *also* **true templating**

- CMS building, 590–591
- ExpressionEngine, 582–583
- MVC design pattern view, 307
- native PHP, 310
- producing email output with, 268–270
- `search.phtml`, MVC mini toolkit, 326–327
- `searchresults.phtml`, MVC mini toolkit, 330
- Zend Framework, 379
- Zend IDE, 650

ten-point plan, 423

test case class, 345

test cases, PHPUnit, 347–350

test suite

- PHPUnit testing, 350
- unit testing, 345

`TestConstraints()` **method, 329**

test-driven development, 425–426

testing

- case study development approach, 388
- customer contact, 465–467
- extreme programming, 426
- fault tracking, 543–544
- functional testing, 541–542
- `HTTPSession` class, 297–300
- load testing, 542
- performance tuning, 667
- PHP installation, 679
- satisfying, 467–469
- Travel Expense Report application. See Travel Expense Report
- unit testing. See unit testing
- usability testing, 543
- using Mantis. See Mantis
- waterfall process vs. spiral, 418
- writing, building application, 452–453

testing phase, 424–425

text message

- notification, 249
- recipients, 251
- validation, 250

themes, Drupal, 577

thin clients, implementing, 274

threading, 601–602

three-tier Ethernet segmentation, 609–610

thttpd Web servers, 599

timelines, receiving formal brief of, 400–401

timestamps, 206–208, 282

time-to-live (TTL) edicts, 604

timezones, PHP, 207

toolbox, Komodo, 656

topology, version control, 641–643

traffic

- determining business requirements, 433
- PHP and. See high traffic/high availability Website, 611–613

transactions, PDO, 142–143

transfer rate, 601

transitions, UML activity diagrams, 42

Travel Expense Report

- more travel expense week tests, 500–502
- overview of, 492–494
- release planning, 450
- satisfying travel expense week tests, 502–514
- the spike, 499–500
- travel expense item, 494–496

travel expense week, 496–499

Travel Expense Report, finalized

- anything else, 526–529
- overview of, 514–520
- as spreadsheet, 520–526

triggered e-mails, providers for, 274

true templating

- advanced Smarty, 340–342
- defined, 327
- installing Smarty, 334–335
- native PHP templating, 333
- with Smarty templates, 334
- using Smarty, 335–340
- when to use traditional vs. Smarty, 342

`try...catch` **blocks**

- exception handling in SOAP client, 240
- handling errors in PDO, 144–145
- wrapping method calls in collections, 124–126

TTL (time-to-live) edicts, 604

U

ubiquity, of PHP, 624

Ulysses

- business objects, 364–365
- comparing Prado Framework to, 377–378
- as favorite framework, 377
- heading home, 374
- installing, 361–363
- introduction to, 361
- persistence, 372–373
- processing user input, 371–372
- quick check, 374
- quick summary, 375–377
- saving input, 374–375
- servlets, 366–371
- setting up, 363–364
- validation, 373–374
- what needs to be done, 432–433
- working with, 363

UML (Unified Markup Language)

- activity diagrams, 42–44
- class diagrams. See class diagrams, UML
- component and deployment diagrams, 47–48
- creating contact manager application, 50–56
- designing `Collection` class, 105–106
- inheritance using, 16–17
- requirements gathering phase of, 31–32
- sequence diagrams, 44–46
- state diagrams, 46–47

UML (Unified Markup Language) (continued)

summary, 48

updating, 64

use case diagrams, 33–35

unidirectional navigability, 37–38

Unified Markup Language. See UML (Unified Markup Language)

Uniform Resource Locators (URLs), and REST, 243–246

unit testing

case study development approach, 391

creating test suite for class, 345

designing interface of class, 344–345

in extreme programming, 426

overview of, 343–344

for quality assurance, 540–541

reasons for using, 351–352

test-driven development using, 426

Travel Expense Report application. See Travel Expense Report

using PHPUnit, 346–350

writing implementation of your class, 346

UNIX servers

choosing, 673–674

defined, 672

excellence as server environment, 672–673

installing PHP on, 673

operating systems for, 598

transferring PHP to Windows from, 679–680

updates

designing widgets, 84–86

implementing business objects, 162, 167–168

observer pattern, 83

URL rewriting

frameworks as mechanism for, 357

session perpetuation using, 279–280

in Ulysses framework, 376

URLs (Uniform Resource Locators), and REST, 243–246

usability of product, QA, 539

usability testing, 543

use case diagrams, 33–35

use cases

application framework scenario, 359–361

sequence diagrams tied to, 44

user agent consistency, 286

user communications

blocking activity, 271

broadcast e-mail providers, 274

as class hierarchy, 251–258

deliverability, 272–273

domain keys and DKIM, 273–274

e-mailing your users. See emailing users

email-to-fax gateways, 271

maintaining job queue, 271–272

reasons for, 247–250

Sender ID, 273

SMS text messaging, 270

SPF records, 273

types of, 250–251

usernames, 277–278, 286

users, Mantis, 548–550

utility classes

business objects vs., 343

frameworks providing, 359

overview of, 156

in Ulysses framework, 377

V

_v flag, rsync, 564

validation

frameworks providing, 357, 358

Ulysses framework and, 373–374, 376

user communication through, 249–250

variable modifiers, Smarty, 341

variables, 287

veneer of product, QA, 539–540

version control

advanced techniques, 645–646

automated. See automated version control, repository extraction

deployment for PHP frameworks, 379–380

FTP and, 567

keeping servers in sync, 566–567

maintaining in build phase, 424

overview of, 635

principles of, 635–641

software, 643–645

topology, 641–643

using `rsync`, 564–567

Zend integration with, 654

view, MVC design pattern

file naming conventions, 310

introducing, 306

overview of, 307–308

in Web applications, 308

View Issues screen, Mantis, 553

virtualization, 597

visibility, member variables, 11, 28

W**waterfall approach**

- choosing, 416–417
- extreme programming vs., 391
- overview of, 389–391

Web applications, MVC in, 308**Web development skills, PHP professionals, 628–629****Web servers**

- for application platforms, 598–599
- determining, 434
- Flickr, 616
- keeping in sync, 566–567
- load, 601–602
- load balancing, 603–605
- PHP installation using, 673–674

Web Service Description Language. See WSDL (Web Service Description Language)**Web services**

- overview of, 225–226
- REST, 243–246
- SOAP. See SOAP (Simple Object Access Protocol)

Widget object

- connecting to Observable DataSource, 88–89
- designing in observer pattern, 84–86
- implementing decorator pattern, 89–91

Widget World software development case study

- background of, 384–386
- building application. See building application
- deployment. See deployment
- financial layer, 386
- political layer, 386
- project management. See project management
- project overview. See project overview
- project planning. See project planning
- systems architecture. See systems architecture
- technical layer, 386
- WW-SFAT requirements, 387
- your role in, 387

Wikipedia, 616–618**Wildcard DNS setups, 556****Windows. See Microsoft Windows****wireframes, information architecture design, 422–423****workflow, in content management, 572, 589–590****working practices, project team, 413****worms, 284****wrapper classes, 133–134****write() function, 291****write-only statements, PDO, 141–142****Writing Compilers and Interpreters (Mak), 630****WSDL (Web Service Description Language)**

- enhancing SOAP application, 227
- making SOAP client, 230–231
- making SOAP server, 240–243
- SOAP making use of, 226
- SoapClient functions, 228
- understanding use of in SOAP, 232–237

WW-SFAT manager, 384–385, 387**X****XML**

- REST using, 244
- SOAP using, 226

XP (extreme programming)

- case study development using, 387–391
- defined, 426
- understanding, 426–427
- when to use, 428

Z**_z flag, rsync, 565****Zend (Zend Studio)**

- debugging with ZDE, 651–654
- editing code, 649–650
- inspecting code, 650–651
- managing projects, 649
- more tools, 654
- overview of, 648–649
- Zend Framework, 378–379
- Zend Guard, 654