

# Index

## A

- `acceptVisitor()`, **123, 124**
- `action()`, **51**
- `activityStream`, **89**
- Adapter**, **25–26, 228**
- Adapter Design Pattern**, **25–30**
  - contact importing, **154**
  - Zend Framework, **14**
- `add()`, **116–117**
  - HTML, **203**
  - Users, **193**
  - UsersModule, **149**
- `addFilter()`, **14**
- `addMethods()`, **222**
- `addmethods()`, **152**
- `addNewItem()`, **89**
- `addObserver()`, **87**
- `addSong()`, **52**
- `addSongs()`, **53**
- `addTrack()`, **46, 76**
- `addValidator()`, **15**
- admin access**, **149–151**
- admin status**, **195**
- AJAX**
  - Proxy Design Pattern, **97**
  - Strategy Design Pattern, **111–112**
- algorithms**
  - Interpreter Design Pattern, **67**
  - Strategy Design Pattern, **110, 111, 114**
  - Visitor Design Pattern, **125**
- Anchor**, **213**
- Apache**, **161**
- application access**, **134**
- application core**, **141–143**
  - programming, **161–175**
- application programming**, **159–226**
  - authentication, **185–187**
  - contact information, **197–226**
  - index, **175–179**
  - login, **175–179**
- `applyAdjustment()`, **116**
- `applyInterpretation()`, **69**
- `$array`, **164, 205, 232**
- ArrayAccess**, **16**
- `array_map()`, **59**
- arrays**, **73**
  - contact information, **231–233**
  - Delegate Design Pattern, **231**
  - importing, **205**
  - Iterator Design Pattern, **78**
  - Mediator Design Pattern, **84**
  - `setitem()`, **178**
  - `$values`, **182**
- `/assets/main.css`, **171**
- `/assets/managecontact.js`, **219**
- `/assets/managecontacts.js`, **216**
- assumptions**, **131, 137**
- `attach()`, **15**
- `attachObserver()`, **88–89**
- Auth**, **178**
- auth**, **147**
  - `authenticate()`, **179**
  - `isLoggedInin()`, **175**
  - `process()`, **185**
  - UML, **174, 179**
- Auth module**, **13**
- `authenticate()`, **148**
  - auth, **179**
  - contactcollection, **179**
  - factory(), **186**
  - username, **179**
- `authenticateusingfactory()`, **147, 148, 186**
- authentication**, **143–148**
  - application programming, **185–187**
  - contactcollection, **148**
  - Factory Design Pattern, **147**
  - `isLoggedInin()`, **175**
  - templates, **175**
  - User, **144, 185**
  - user, **178**

**authenticatorinterface**, 186  
**auth::isAdmin()**, 179  
**auth::isLoggedIn()**, 179  
**authorization**, 143–148  
**authstandard**, 148, 186  
**autoloader.php**, 162–163

## B

**\$band**, 88  
  Prototype Design Pattern, 94  
  Proxy Design Pattern, 100  
  Strategy Design Pattern, 112  
  Visitor Design Pattern, 124  
**BandEndorsedCaseOfCereal**, 119  
**\$bandMixProto**, 95  
**\$bandName**, 77  
**banking**, **Template Design Pattern**, 116–117  
**baseDAO**, 39, 41  
**Blog**, 116  
**blogs**  
  Decorator Design Pattern, 44  
  Factory Design Pattern, 62  
  Observer Design Pattern, 87  
  Visitor Design Pattern, 122  
**Boerger, Marcus**, 15  
**\$boughtCDs**, 107  
**build()**, 34–35  
**buildarray()**, 205, 231  
**buildcollection()**, 153, 204, 232  
**Builder**, 32–33  
**Builder Design Pattern**, 31–35, 153, 204  
**buildMyObject()**, 33  
**buy()**, 95, 100, 107, 124  
**buyCDNotifyStreamObserver**, 89

## C

**C, Design Patterns**, 7  
**Cache module**, 14  
**cache.php**, 14  
**caching**  
  Observer Design Pattern, 87  
  Proxy Design Pattern, 98  
**callRequiredLogic()**, 57  
**CD**, 46  
  Façade Design Pattern, 57–59  
  Factory Design Pattern, 63–65  
  Iterator Design Pattern, 76–78  
  Mediator Design Pattern, 81–84  
  Observer Design Pattern, 88–89  
  Prototype Design Pattern, 93–95  
  Proxy Design Pattern, 99–101  
  SaleItemTemplate, 119  
  Singleton Design Pattern, 107  
  Strategy Design Pattern, 111  
  Template Design Pattern, 118–119  
  Visitor Design Pattern, 123–125  
  XML, 58  
**\$cd**, 95, 114, 124  
**\$\_CD**, 47  
**CDAsJSONStrategy**, 114  
**CDAsXMLStrategy**, 114  
**CDFactory**, 65  
**\$cdID**, 78  
**CDs**. *See compact discs*  
**\$\_CDs**, 77  
**CDSearchByBandIterator**, 77–78  
**CDTrackListDecoratorCaps**, 47  
**CDUpperCase**, 59  
**CDusesStrategy**, 113  
**CDusesStrategyget()**, 114  
**CDVisitorPopulateDiscountList**, 125  
**change()**, 83  
**changeBandName()**, 82, 83  
**changeIdentifier()**, 81  
**Checking**, 116  
**child**. *See parent-child relationship*  
**class extensions**, 44  
**classes**  
  Factory Design Pattern, 61–62  
  \_get(), 182  
  Interpreter Design Pattern, 67  
  Iterator Design Pattern, 74  
  Mediator Design Pattern, 82  
  Observer Design Pattern, 86  
  \_set(), 182  
  Singleton Design Pattern, 106  
  SQL, 181  
  Strategy Design Pattern, 110  
  Template Design Pattern, 116, 117, 119–120  
  Visitor Design Pattern, 123  
**class\_exists()**, 166  
**classname**, 21  
**\$classname**, 232  
**clean()**, 181, 183, 184  
  \_clone(), 94  
**ClonedObject**, 93  
**code**  
  Adapter Design Pattern, 27–30

- Builder Design Pattern, 34–35
- Data Access Object Pattern, 39–42
- Decorator Design Pattern, 45–47
- Delegate Design Pattern, 51–54
- Design Patterns, 7
- Façade Design Pattern, 57–59
- Factory Design Pattern, 63–65
- Interpreter Design Pattern, 69–72
- Iterator Design Pattern, 75–78
- Mediator Design Pattern, 81–84
- Observer Design Pattern, 87–89
- Prototype Design Pattern, 91, 93–95
- Proxy Design Pattern, 99–101
- Singleton Design Pattern, 105–107
- Strategy Design Pattern, 111–113
- Template Design Pattern, 117–120
- Visitor Design Pattern, 123–125
- code snippets**
  - dao, 181
  - Eclipse PDT, 17–21
  - templates, 20
- collection, 209**
- Community Library, 92**
- compact discs (CDs), 45–47**
  - Interpreter Design Pattern, 69–72
  - Iterator Design Pattern, 75–78
  - Mediator Design Pattern, 81–84
  - Observer Design Pattern, 87–89
  - Prototype Design Pattern, 93–95
  - Proxy Design Pattern, 99–101
  - Singleton Design Pattern, 105
  - Visitor Design Pattern, 123–125
- complexFunctionA(), 33**
- complexFunctionB(), 33**
- conditional statements**
  - dao, 181
  - Delegate Design Pattern, 50
- Config, 14**
- Config module, 14**
- configurationOptions, 33**
- connect(), 14, 100**
- \_connect(), Proxy Design Pattern, 100–101**
- \_connectionToDB(), 41**
- constraints, 131, 137**
- \_construct(), 47, 142, 146, 150, 165**
  - contactsCollection, 200
  - Iterator Design Pattern, 77
  - Singleton Design Pattern, 106
  - \$urlstring, 166
- Contact, 152**
- contact**
  - \$contact, 210
  - contactgroup, 206, 209
  - id, 216
  - save(), 205
  - User, 198
- \$contact, 210**
- contact information, 131–132, 133–134, 137–138**
  - adding, 214–223
  - administration, 151–157
  - application programming, 197–226
  - arrays, 231–233
  - Design Patterns, 237
  - editing, 223–225
  - importing, 153–154, 202–209, 227–233
  - relationships, 153
  - viewing, 155–157
- contact sync, 139**
- contactcollection, 144**
  - authenticate(), 179
  - authentication, 148
  - daocollection, 199–200
  - defaultactation(), 175
  - Index, 187
  - usercollection, 155
- contactgroup, 153, 154**
  - contact, 206, 209
  - contactmethod, 198, 206, 217, 223
  - contactmethodcollection, 216
  - id, 223
  - NULL, 206–207
  - populate(), 210
- contactgroupcontainer, 216**
- contactgroupid, 156**
- contactgrouping, 216**
- contactgroupsCollection, 234–235**
- contactid, 156**
- contactimportgroupinterpreter, 206**
- contactmethod, 153**
  - contactgroup, 198, 206, 217, 223
  - dao, 156
- contactmethodscollection, 156, 235–236**
- contactmodule, 155**
- Contacts, 188**
- contacts**
  - addMethods(), 222
  - edit(), 223
  - processadd(), 221

- contacts (continued)**
  - `processedit()`, 224
  - `processimport()`, 205, 229
- contacts sync, 132, 135**
- /contacts/add, 214**
- contacts/browse, 200, 201**
- contactscollection, 152, 155, 200**
  - `getwithdata()`, 200
  - looping, 201
- contacts/edit, 224**
- 'contactsfile', 229**
- contacts/group, 216**
- /contacts/import.php, 202**
- contacts/manage, 214, 219**
- ContactsModule**
  - Module, 152
  - MySQL, 198
  - `view()`, 152, 157
- contacts/processadd, 221**
- /contacts/view, 209**
- contacts/viewsidebar, 211**
- content, 69**
- \$content, 166, 169**
- content.applyInterpretation(), 69**
- content-filtering proxy, 98**
- contents, 231**
- \$contents, 233**
- Controller, 142, 144**
- controller**
  - `edit()`, 195
  - `index.php`, 163, 169
  - `render()`, 169
- Coordinated Universal Time (UTC), 122**
- create, read, update, delete applications (CRUD), 37**
- create(), 59**
  - databases, 145
  - Factory Design Pattern, 65
  - `save()`, 145, 182
  - `$table`, 182
  - Template Design Pattern, 116
  - `UsersModule`, 149
- Create Category, 19–20**
- createObject(), 63**
- credentials, 98, 136**
- CRUD. See create, read, update, delete applications**
- CSS, 166, 169, 170**
- CSV**
  - Adapter Design Pattern, 26

- `errorObject`, 28
- .csv, 153**
  - Input, 228
  - Outlook, 154, 159–160, 202
  - `processimport()`, 204
- current(), 16, 17–18, 76**
  - `daocollection`, 150
  - Iterator, 189
  - public methods, 189
- Customize Palette, 19**

## D

**DAO. See Data Access Object**

- dao, 149**
  - code snippets, 181
  - conditional statements, 181
  - `contactmethod`, 156
  - `dataobjects`, 162
  - NULL, 181
  - parent-child relationship, 181
  - `populate()`, 181, 182
  - `property()`, 181
  - `save()`, 145
  - UML, 181
  - User, 145, 152, 179–182, 186
  - user, 178, 179

- daocollection, 155, 156**
  - `contactcollection`, 199–200
  - `contactmethodscollection`, 156
  - `getwithdata()`, 189
  - Iterator, 150, 156
  - Iterator Design Pattern, 149
  - parent-child relationship, 150, 189
  - `populate()`, 189, 208
  - Template Design Pattern, 149, 150
  - UML, 188
  - `usercollection`, 199–200
  - `UsersCollection`, 188

**daocollectioninterface, 156**

**Data Access Object (DAO), 38, 147**

**Data Access Object Design Pattern, 15, 37–42, 145**

**'DATA TRACK', 64**

- databases**
  - abstraction, 15
  - connections, 104
  - `create()`, 145
  - Template Design Pattern, 145
  - `update()`, 145

**DataObject, 13**  
**dataobjects, 162**  
 /dataobjects/contactgroup.php, 198  
 /dataobjects/contactmethod.php, 198  
 /dataobjects/user.php, 182  
**db(), 5**  
**db, 145**  
   clean(), 181, 183  
   execute(), 183  
   factory(), 184  
   Factory Design Pattern, 181  
   getArray(), 150, 183  
   insertGetID(), 183  
   MySQL, 181  
   public methods, 184  
   UML, 182  
**Db module, 14**  
**DB2, 14, 103**  
**dbConnection, 39**  
**decorate(), 212**  
**decorateItems(), 45**  
**Decorator Design Pattern, 43–47, 156–157**  
**decoratorinterface, 157**  
**decoratorsocialnetwork, 213**  
**default(), 144**  
**defaultaction, 166**  
**defaultaction()**  
   contactcollection, 175  
   Index, 187  
   index, 199  
   login, 176  
   show(), 191  
**\$delegate, 232**  
**Delegate Design Pattern, 49–54, 231**  
   setviewtype(), 237  
**delegateType, 51**  
**delete(), 116, 197**  
**Delete user, 192**  
**"deletecontactgrouping", 221**  
**deleteMethods(), 225**  
**deletemethods(), 152**  
**Design Patterns, 3–9. See also Specific Design Patterns**  
   C, 7  
   choosing and planning, 141–158  
   code, 7  
   contact information, 237  
   deleting objects, 238  
   error checking, 237

  Java, 7  
   name, 6  
   PEAR, 12  
   plug and play, 7, 13  
   problems, 6  
   refactoring, 8  
   solutions, 6  
   templates, 5  
   Zend Framework, 13  
**detach(), 15**  
**detailed requirements, 137–138**  
**DirectoryIterator, 16**  
**Div, 216, 219–221**  
   .dmg, 50  
   doCalculation(), 117  
**Doctrine, 15**  
**DomDocument, 112**  
**doSomething(), 27, 87, 99**  
**doSomethingA(), 57**  
**doSomethingB(), 57**  
**doSomethingRequiresAandB(), 57**

## E

**Eclipse PDT, 11, 17–21**  
**e-commerce**  
   Observer Design Pattern, 86–87  
   Template Design Pattern, 117–120  
**edit()**  
   contacts, 223  
   controller, 195  
   Façade, 237  
   users, 195  
   UsersModule, 149  
**else, 52**  
**end(), 142, 168**  
**enhancedCD, 64–65**  
**Entity, 116**  
**error checking, 237**  
**errorObject, 27–30**  
**execute(), 145**  
   db, 183  
   insertGetID(), 184  
   \$query, 184  
**executive summary, 136–137**  
**\$externalPurchaseInfoBandID, 95**  
**\$externalRetrievedType, 52**

## F

**Façade, 59, 237**

**Façade Design Pattern, 55–59**

**facadecontactinformation, 234**

**Factory, 62, 65**

**factory(), 14, 145, 147**

authenticate(), 186

authstandard, 186

db, 184

getInstance(), 183

getInstance, 184

importadapter, 228, 230

LDAP, 186

**Factory Design Pattern, 61–65,**

**183, 229**

authentication, 147

db, 181

MySQL, 145, 181

PEAR Mail, 12

PEAR MDB2, 12

Zend Framework, 13–14

**fetch(), 39, 41**

**\$fgroupname, 206**

**File Input, 229**

**file systems**

Iterator Design Pattern, 74

looping, 74

Prototype Design Pattern, 92

**Filter.php, 14**

**findviewtype(), 143**

**firstname, 154**

**Form module, 14**

**formid, 214**

## G

**Gamma, Erich, 5**

**generateimportmethods(), 207, 208**

**generatePublicly(), 116**

**generateZip(), 50**

**get(), 113–114**

**\_get(), 145, 182**

**getAddressByZip(), 38**

**getArray(), 145**

\$array, 232

contents, 231

db, 150, 183

MySQL, 184

\$return, 184

**getAsXML(), 112–113**

**getBuiltMyObject(), 33**

**getChildren(), 16**

**getContent(), 69**

**getErrorNumber(), 28–29**

**getErrorText(), 28–29**

**getgroup(), 154, 206**

**getGuitar(), 62**

**getHttpClient(), 13**

**getInstance(), 13, 183, 184**

mysql, 145

Registry, 13

singleton, 184

Singleton Design Pattern, 105, 106, 107

\$type, 183

**getinstance(), 184**

**getitem(), 164, 177**

**getM3U(), 52–53**

**getMyObjects(), 75**

**getName(), 75**

**getNext(), 75**

**getPlaylist(), 53–54**

**getPLS(), 52–53**

**getProduct(), 34**

**getProfilePage(), 70**

**getTitle(), 70**

**getTrackList(), 46**

**\$GET['u'], 162**

**\$\_GET['u'], 164**

**getUserByFirstName(), 41**

**getwithdata(), 150, 155, 207**

contactmethodscollection,

235–236

contactscollection, 200

daocollection, 189

generateimportmethods(), 208

Template Design Pattern, 189

userscollection, 191

**Globally Unique Identifier (GUID), 92**

**\$GLOBALS, 104**

**Google, 56**

**Google Maps, 212**

**group**

NULL, 216

/views/default/contacts/group

.php, 216

**groupcount, 220**

**groups, 234**

**GUID. See Globally Unique Identifier**

**Guitars, 62**

**H****\$\_handle**, 106**handlers**, 12**hasChildren()**, 16**hash tables**, 73**hasNext()**, 75**Helm, Richard**, 5**.HTACCESS**, 161–162**.htaccess**, 162**.htaccess**, \$\_GET['u'], 164**HTML**, 166

add(), 203

Div, 216, 219–221

id, 224

Input, 177, 216, 219–221, 223

Interpreter Design Pattern, 68

Select, 219–221, 223

Span, 219–221

**.html**, 9**HTTP**, 176**HTTP User Agent**, 166**I****id**

contact, 216

contactgroup, 223

contactgroupcontainer, 216

HTML, 224

JavaScript, 216

**\$id**, 236**identifier**, 81**if/else**, 52–53, 62**import()**, 202**importadapter**

factory(), 228, 230

setcontents(), 230

**importcontactarraybuilder**, 233**importcontactarrayinterface**, 231**importcontactsarraybuilder**, 204, 231**importcontactscollectionbuilder**, 152**importedstring**, 153**importing**

arrays, 205

Builder Design Pattern, 204

contact information, 153–154, 202–209,  
227–233

Interpreter Design Pattern, 206

Outlook, 227–228

**\$imports**, 204, 205

'importtype', 229

/includes/authenticatorinterface  
.php, 186

/includes/auth.php, 185

includesautoload(), 164

/includes/contactgroupscollection  
.php, 209/includes/contactscollection  
.php, 199

/includes/controller.php, 164

/includes/daocollection.php, 188

/includes/dao.php, 179

/includes/db.php, 182

/includes/decoratoraddress  
.php, 212/includes/decoratorinterface  
.php, 212/includes/decoratormobilephone  
.php, 213/includes/decoratorsocialnetwork  
.php, 213

/includes/decoratorwebsite.php, 213

/includes/exceptions.php, 166

/includes/importadapter.php, 230

/includes/importcontactsarraybuilder  
.php, 204, 232

/includes/lib.php, 163

/includes/mysql..php, 183

/includes/outlookcontactimportadapter  
.php, 205, 230/includes/outlookimportcontact  
sarraydelegate.php, 232/includes/singletoninterface  
.php, 184**Index**, 187**index**, 175–179, 199**indexcontacts**, 209**IndexModule**, 144, 155**index.php**, 162–166, 169

controller, 163, 169

.htaaccess, 162

view, 166

**InfoCard**, 14**InfoCard.php**, 14**initial contact import**, 138, 139**initial requirements analysis**, 132–136**Input**

.csv, 228

HTML, 177, 216, 219–221, 223

- insertGetID()**, 145, 183, 184
- instance**, 105
- \$instance**, 184
- \$\_instance**, 106
- internalDelegate**, 51
- InternalException**, 166
- Internet Explorer**, 134–135
- Interpreter Design Pattern**, 67–72, 237
  - importing, 206
- interpretKeys()**, 69
- inventory system**, Factory Design Pattern, 62
- InventoryConnection**, 106–107
- isAdmin()**, 148, 185
- isDot()**, 16
- isLink()**, 16
- isloggedin()**, 148, 175
  - Login, 185
- is\_uploaded\_file()**, 204
- isValid()**, 14
- Iterator**, 14, 16
  - current(), 189
  - daocollection, 150, 156
  - Eclipse PDT, 17–18
  - key(), 189
  - next(), 189
  - rewind(), 189
  - valid(), 189
- Iterator Design Pattern**, 73–78
  - daocollection, 149
  - Zend Framework, 14
- iterator.php**, 12

## J

- Java**, 7
- JavaScript**, 169, 170
  - /assets/managecontacts.js, 216
  - id, 216
  - Strategy Design Pattern, 111
- JavaScript Object Notation (JSON)**, 112
- Johnson, Ralph**, 5
- Join**, 15
- JSON**. See JavaScript Object Notation

## K

- Kerievsky, Joshua**, 8
- key()**, 16, 17, 189
  - daocollection, 150

- Eclipse PDT, 17–18
- Iterator Design Pattern, 76
- keyedUpdateObject**, 39
- keyItem**, 39
- \$keys**, 205
- keywords**, 68

## L

- lastname**, 154
- LDAP**, 136, 186
- lib**, 163
  - getitem(), 177
  - makehashedpassword(), 187
  - sendto(), 176
  - setError(), 178
  - setitem(), 164
- lib:getitem()**, 177
- lib::setError()**, 178
- lib::setitem()**, 177
- LimitIterator**, 16
- \$link**, 184
- living requirements document**, 43
- Log()**, 14
- logicA()**, 117
- LogicFacade**, 57
- Login**, 179, 185
- login**
  - application programming, 175–179
  - defaultaction(), 176
  - process(), 178
  - public methods, 178
- LoginModule**, 144
- /login/process**, 177
- \$logline**, 124
- Logout**, 187
- LogoutModule**, 148
- Log.php**, 13
- logToCSV**, 28, 30
- looping**, 73
  - contactscollection, 201
  - file systems, 74
  - MySQL, 74

## M

- M3U**, 51
- macro language**, 67
- mail**, 12

- Mail.php, 12**
  - main filter(), 14**
  - main views, 169–175**
  - makeArray(), 59**
  - makeCaps(), 47**
  - makehashedpassword(), 187**
  - makeString(), 59**
  - makeXMLCall(), 59**
  - mbd2.php, 12**
  - \$\_mediator, 83**
  - Mediator Design Pattern, 79–84**
    - deleting objects, 238
  - methodA(), 27**
  - methodB(), 27**
  - methodboxvaluebox, 219**
  - method\_exists(), 166**
  - \$methods, 236**
  - \$methodtypekey, 223**
  - middleware, 154**
  - MixtapeCD, 94–95**
  - mobile access, 134**
  - mock**
    - Interpreter Design Pattern, 70
    - PEAR Mail, 12
  - Model-View-Controller (MVC), 142**
  - mod\_rewrite, 162**
  - Module, 142, 144, 152**
  - moduleautoloader(), 163**
  - /modules/contacts.php, 202**
  - /modules/login.php, 176**
  - /modules/users.php, 189**
  - MP3, 51–53**
    - Mediator Design Pattern, 81–84
  - MP3Archive, 83**
  - MVC. See Model-View-Controller**
  - \$myCDCaps, 47**
  - {myCD.getTitle()}, 70**
  - MyDelegateObject(), 51**
  - MyDelegateObject, 51**
  - MyInterpreter, 69**
  - MyObject(), 105**
  - MyObject**
    - Builder Design Pattern, 33
    - Decorator Design Pattern, 45
    - Delegate Design Pattern, 51
    - Facade Design Pattern, 57
    - Interpreter Design Pattern, 69
    - Iterator Design Pattern, 75
    - Observer Design Pattern, 87
    - Prototype Design Pattern, 93
    - Proxy Design Pattern, 99
    - Singleton Design Pattern, 105
    - Strategy Design Pattern, 111
    - Visitor Design Pattern, 123
  - MyObjectA, 81**
  - MyObjectAdapterForNewConsumer, 27**
  - MyObjectB, 81**
  - MyObjectBuilder, 33**
  - MyObject.change(), 111**
  - myObjectChanged(), 81**
  - MyObjectCollection, 75**
  - MyObjectCollectionIterator, 75**
  - MyObjectDecorator, 45**
  - MyObjectDecorator(), 45**
  - MyObjectFactory, 63**
  - MyObjectMediator, 81**
  - MyObjectsToMediate, 81**
  - MyObjectStrategy, 111**
  - MyObjectTypeA, 63**
  - MyObjectTypeB, 63**
  - MyProxyObject, 99**
  - MySQL, 14, 161**
    - Adapter Design Pattern, 26
    - clean(), 184
    - ContactsModule, 198
    - Data Access Object Design Pattern, 39–42
    - db, 181
    - deleteMethods(), 225
    - Factory Design Pattern, 145, 181
    - getArray(), 184
    - Iterator Design Pattern, 74, 75–78
    - looping, 74
    - Prototype Design Pattern, 92
    - Proxy Design Pattern, 100
    - Singleton Design Pattern, 103, 105
    - \$user, 200
  - mysql, 225**
    - getInstance(), 145
    - getinstance(), 184
    - \$link, 184
    - Singleton Design Pattern, 145
    - UML, 182
  - mysql\_insert\_id(), 184**
  - MyTemplate, 117**
  - MyVisitor, 123**
- ## N
- name, 75, 229**
  - \$name, 164**

**newID**, 81  
**newPlaylist**, 53  
**\$newValue**, 84  
**next()**, 16, 17

- daocollection, 150
- Eclipse PDT, 17–18
- Iterator, 189
- Iterator Design Pattern, 76, 78
- public methods, 189

**notify()**, 15, 87  
**notifyMediator()**, 81  
**notifyObserver()**, 88  
**NULL**, 32, 164

- contactgroup, 206–207
- dao, 181
- getitem(), 177
- group, 216
- Singleton Design Pattern, 105

**null**, 83

## O

**Object Oriented Programming**, 5

- Delegate Design Pattern, 49
- Mediator Design Pattern, 79
- parent-child relationship, 43
- Prototype Design Pattern, 91
- Singleton Design Pattern, 104
- Strategy Design Pattern, 109–110

**object relational mapper (ORM)**, 15  
**Observer Design Pattern**, 13, 85–89

- Zend Framework, 14–15

**observers**, 87  
**\$\_observers**, 88  
**ob\_start()**, 142, 166  
**open source software**, 85

- Visitor Design Pattern, 122

**Oracle**, 14, 26  
**OriginalConsumer**, 27  
**\$originalObject**, 84  
**ORM**. See **object relational mapper**  
**Outlook**, 135, 138

- .csv, 154, 159–160, 202
- importing, 227–228
- 'outlook', 229

**outlookcontactimportadapter**, 154, 230  
**outlookcontactcollectionbuilder**, 154  
**outlookcontactimportadapter**, 205, 228  
**output buffering**, 170  
**oversizedAddition()**, 118–119

## P

**parent-child relationship**

- DAO, 38
- dao, 181
- daocollection, 150, 189
- Object Oriented Programming, 43
- sav(), 145
- Strategy Design Pattern, 110
- Template Design Pattern, 116

**\$parts**, 166  
**passwords**, 144, 147, 195

- SHA-1, 187
- username, 178

**PEAR**, 6, 11, 12  
**PEAR DB**, 13  
**PEAR Log**, 13  
**PEAR Mail**, 12

- Factory Design Pattern, 12
- handlers, 12
- mock, 12

**PEAR MDB2**, 12–13  
**\$persist**, 164  
**.php**, 9  
**PHP Extension of Application**.  
See **PEAR**  
**Playlist**, 51–54  
**\$playlistContent**, 52  
**PLS**, 51  
**plug and play**, 7, 13  
**plugin system**, 86  
**populate()**, 145, 150

- contactgroup, 210
- dao, 181, 182
- daocollection, 189, 208
- \$table, 182

**\_populateDiscountList()**, 125  
**populategroups()**, 234  
**populatemethods()**, 235, 236  
**position**, 150  
**\$price**, 118, 120, 124  
**problems**

- Adapter Design Pattern, 25–26
- Builder Design Pattern, 31–33
- Decorator Design Pattern, 43–45
- Delegate Design Pattern, 49–50
- Design Patterns, 6
- Façade Design Pattern, 55–56
- Factory Design Pattern, 61–62
- Interpreter Design Pattern, 67–68

- Iterator Design Pattern, 73–74
- Mediator Design Pattern, 79–80
- Observer Design Pattern, 86–87
- Prototype Design Pattern, 91–93
- Proxy Design Pattern, 97–99
- Singleton Design Pattern, 103–104
- Strategy Design Pattern, 109–111
- Template Design Pattern, 115–117
- Visitor Design Pattern, 121–122
- process(), 144**
  - Auth, 178
  - auth, 185
  - lib::setitem(), 177
  - login, 178
  - /login/process, 177
  - User, 178
- processadd(), 149, 156, 193, 224**
  - contacts, 221
  - standard/errors, 194
  - users, 195
- processdelete(), 149, 156, 197**
- processedit(), 149, 156, 194, 195, 224**
- processimport(), 153, 202, 205, 229**
  - .csv, 204
- processingimport(), 204**
- product, 34–35**
- productBuilder, 34–35**
- Profile, 111**
- property(), 181**
- protected method**
  - Singleton Design Pattern, 104, 105
  - Visitor Design Pattern, 122
- Prototype Design Pattern, 91–95**
- provideProxyFeature(), 99**
- Proxy, 98**
- Proxy Design Pattern, 97–101**
- public methods, 144**
  - current(), 189
  - db, 184
  - isAdmin(), 185
  - key(), 189
  - login, 178
  - next(), 189
  - rewind(), 189
  - saveall(), 189
  - Template Design Pattern, 116
  - valid(), 189

## Q

- \$query, 184**

## R

- RDBMS. See relational database management system**

- RecursiveIterator, 16**

- refactoring**

- Design Patterns, 8
- Observer Design Pattern, 86
- Prototype Design Pattern, 92
- Strategy Design Pattern, 109

- Registry, 13**

- Registry.php, 13**

- regression, Adapter Design Pattern, 26**

- relational database management system (RDBMS), 12**

- removal, 192**

- removal.js, 211**

- render(), 142, 165**

- class\_exist(), 166
- Controller, 144
- controller, 169
- view, 166

- \$replacement, 71**

- requestClone(), 93**

- require(), 163**

- requirements analysis, 129–139**

- REST API, 62**

- retrievedAdjustmentOnBalance(), 116**

- \$return, 184**

- rewind(), 16, 17**

- daocollection, 150
- Eclipse PDT, 17–18
- Iterator, 189
- Iterator Design Pattern, 76, 78
- public methods, 189

- RSS feeds**

- Factory Design Pattern, 62
- Observer Design Pattern, 87

- runDelegateAction(), 51**

## S

- SaleItemTemplate, 118, 119**

- save(), 82, 145**

- contact, 205
- create(), 145, 182

### **save() (continued)**

- dao, 145
- update(), 145, 197
- UsersModule, 149
- saveall(), 189**
- scope creep, 79**
- scope items, 131, 137**
- Search Façade, 56**
- searchBySpecificKey(), 39**
- security**
  - Interpreter Design Pattern, 67
  - Singleton Design Pattern, 104
- Seek(), 16**
- SeekableIterator, 16**
- Select, 15, 219–221, 223**
- select, 41**
- sendto()**
  - lib, 176
  - URLs, 176
- \$\_SERVER['DOCUMENT\_ROOT'], 161**
- \$\_SESSION, 164**
- \_set(), 145, 182**
- setAdapter(), 14**
- setColor(), 34**
- setContentts(), 228**
  - \$contents, 233
  - importadapter, 230
- setDatabase(), 32**
- setDelegateType(), 51**
- setError(), 178**
- setHostName(), 32**
- setHttpClient(), 13**
- setItem(), 164, 178**
- setPassword(), 32**
- setPriceAdjustment(), 118**
- setPriceAdjustments(), 120**
- setSize(), 34**
- setStrategyContext(), 113–114**
- setType(), 34**
- setUser(), 70–71**
- setUsername(), 32**
- setviewtype(), 166**
  - Delegate Design Pattern, 237
  - show(), 169
- SHA-1, 187**
- share(), 116**
- shell, 169**
- shell.php, 170**
- show(), 142–143**
  - defaultaction(), 191
  - end(), 168
  - setviewtype(), 169
  - start(), 168
  - UsersModule, 149
  - \$view, 170
  - \$viewtype, 166, 168
- showItemsFormatted(), 45**
- showLogin(), 144**
- SimpleXMLElement, 17**
- SimpleXMLIterator, 17**
- Single Sign-On, 144, 147**
- singleton, 182, 184**
- Singleton Design Pattern, 103–107**
  - mysql, 145
  - Zend Framework, 13
- SMTP, 12**
- social networking**
  - decorators, 213
  - Iterator Design Pattern, 74
  - Observer Design Pattern, 87–89
  - Proxy Design Pattern, 98–99
  - Template Design Pattern, 116
- solutions**
  - Adapter Design Pattern, 25–26
  - Builder Design Pattern, 31–33
  - Decorator Design Pattern, 43–45
  - Delegate Design Pattern, 49–50
  - Design Patterns, 6
  - Façade Design Pattern, 55–56
  - Factory Design Pattern, 61–62
  - Interpreter Design Pattern, 67–68
  - Iterator Design Pattern, 73–74
  - Mediator Design Pattern, 79–80
  - Observer Design Pattern, 86–87
  - Prototype Design Pattern, 91–93
  - Proxy Design Pattern, 97–99
  - Singleton Design Pattern, 103–104
  - Strategy Design Pattern, 109–111
  - Template Design Pattern, 115–117
  - Visitor Design Pattern, 121–122
- Span, 219–221**
- SPL. See Standard PHP Library**
- spl\_autoload\_register(), 163**
- SplFileInfo, 16**
- SplObserver, 15**
- SplSubject, 15**
- SQL**
  - classes, 181
  - Data Access Object Design Pattern, 37–38

- Doctrine, 15
- Visitor Design Pattern, 125
- stakeholders, 132, 134**
- Standard PHP Library (SPL), 11, 13, 15–17, 74**
- standardCD, 65**
- standard/errors, 194**
- start(), 142**
  - ob\_start(), 166
  - show(), 168
- state change, 86**
- storage, 150**
- storeContent(), 69**
- Strategy, 110–114**
- \$\_strategy, 113**
- Strategy Design Pattern, 109–114**
- \$strategyObject, 113**
- strtoupper(), 47, 59**
- subtract(), 116–117**
- switch/case, 62**
- symbols, Interpreter Design Pattern, 68**

## T

- table, 145**
- \$table**
  - create(), 182
  - populate(), 182
  - update(), 182
- tableName, 39**
- tableNameDAO, 39**
- Tarred and GZipped (TGZ), 122**
- taxAddition(), 118–119**
- Template, 115–120**
- Template Design Pattern, 115–120**
  - daoCollection, 149, 150
  - databases, 145
  - getwithdata(), 189
- templates**
  - authentication, 175
  - code snippets, 20
  - Design Patterns, 5
  - Interpreter Design Pattern, 67, 68
  - isLoggedIn(), 175
  - variables, 20
- TGZ. See Tarred and GZipped**
- .tgz, 50**
- third-parties**
  - Facade Design Pattern, 56
  - Interpreter Design Pattern, 68

- \$this, 124, 189**
- \$this->contents, 233**
- \$this->importedstring, 233**
- \$title, 88**
  - Prototype Design Pattern, 94
  - Proxy Design Pattern, 100
  - Strategy Design Pattern, 112
  - Visitor Design Pattern, 124
- \$track, 76**
- trackList, 46**
- \$trackList, 46, 94**
- TV, Proxy Design Pattern, 98**
- txt, 74**
- type, 63, 142, 214**
- \$type, 53**
  - buildcollection(), 232
  - Facade Design Pattern, 59
  - getInstance(), 183
  - Observer Design Pattern, 88
- 'type', 228**
- 'types', 223**

## U

- UML. See Unified Modeling Language**
- uncoupled objects**
  - Mediator Design Pattern, 84
  - Observer Design Pattern, 85
- Unified Modeling Language (UML), 6**
  - Adapter Design Pattern, 27
  - auth, 174, 179
  - Builder Design Pattern, 33
  - contact viewing, 157
  - dao, 181
  - daoCollection, 188
  - Data Access Object Design Pattern, 38–39
  - db, 182
  - Decorator Design Pattern, 45
  - Delegate Design Pattern, 50–51
  - diagram, 6–7
  - Facade Design Pattern, 56–57
  - Factory Design Pattern, 62–63
  - Interpreter Design Pattern, 69
  - Iterator Design Pattern, 75
  - Mediator Design Pattern, 80–81
  - mysql, 182
  - Observer Design Pattern, 87
  - Prototype Design Pattern, 93
  - Proxy Design Pattern, 99
  - singleton, 182, 184

## Unified Modeling Language (UML) *(continued)*

- Singleton Design Pattern, 105
- Strategy Design Pattern, 111
- Template Design Pattern, 117
- Visitor Design Pattern, 123

**Union, 15**

**unit tests, Visitor Design Pattern, 122**

**United Parcel Service (UPS), 110**

**update(), 39, 41**

- databases, 145
- Observer Design Pattern, 88–89
- save(), 145, 197
- \$table, 182
- UsersModule, 149

**updated requirements**

- discussion, 138–139
- document, 136–138

**updateQuantity(), 106, 107**

**UPS. See United Parcel Service**

**Uri module, 14**

**Uri.php, 14**

**URLs**

- .HTAACCESS, 161–162
- HTTP, 176
- sendto(), 176

**\$urlstring, \_construct(), 166**

**User, 147**

- authentication, 144, 185
- contact, 198
- dao, 145, 152, 179–182, 186
- Data Access Object Design Pattern, 145
- Interpreter Design Pattern, 69–70
- Login, 179
- process(), 178

**user**

- authentication, 178
- dao, 178, 179
- isadmin(), 185
- \$username, 178, 195

**\$user, 200**

**user interface**

- designing, 143–151
- Facade Design Pattern, 56

**userAddress, 38**

**userCD, 71**

**userCDInterpreter, 71**

**usercollection, 155, 199–200**

**username, 195**

- authenticate(), 179
- passwords, 178

- \$username, 178, 195
- \$\_username, 70

**users**

- admin access, 149–151
- creating, 148–149
- deleting, 148–149
- editing, 148–149

**Users, add(), 193**

**users**

- edit(), 195
- processadd(), 195
- processdelete(), 197
- processedit(), 195

**UsersCollection, 149, 188–197**

- daocollection, 188

**userscollection, getwithdata(), 191**

**users/edit, 195**

**users/manage, 193, 195, 214**

**UsersModule, 148–149, 188–197**

- add(), 149
- create(), 149
- edit(), 149
- save(), 149
- show(), 149
- update(), 149

**userTable, 41**

**USPS, 110**

**UTC. See Coordinated Universal Time**

## V

**valid(), 16, 17**

- daocollection, 150
- Eclipse PDT, 17–18
- Iterator, 189
- Iterator Design Pattern, 76, 78
- public methods, 189

**\$\_valid, 77–78**

**Validate.php, 15**

**\$value, 41**

**\$values, 182**

**view, 166–169**

**view()**

- contactmodule, 155
- ContactsModule, 152, 157
- /contacts/view, 209

**\$view, 170**

**\$view['body'], 170**

**\$view['groups'], 211**

/views/default/contacts/browse.php, 200  
 /views/default/contacts/group.php, 216  
 /views/default/contacts/import.php, 203  
 /views/default/contacts/manage.php, 214  
 /views/default/contacts/method.php, 216  
 /views/default/contacts/view.php, 210  
 /views/default/contacts/viewsidebar.php, 211  
 /views/default/index/welcome.php, 201  
 /views/default/login/form.php, 176  
 /views/default/standard/header.php, 174–175  
 /views/default/users/manage.php, 193  
 /views/default/users/show.php, 191  
 /views/default/users/add.php, 193  
 \$viewtype, 166, 168  
 visit(), 123  
 visitCD(), 124  
 Visitor, 123  
 \$visitor, 124  
 Visitor Design Pattern, 121–125  
 Vlissides, John, 5

## W

WebServiceFacade, 59  
 website availability, 131, 137  
 Windows Mobile, 134–135, 212  
 WTAI, 212

## X

### XML

Adapter Design Pattern, 26  
 CD, 58  
 Façade Design Pattern, 57  
 Observer Design Pattern, 89  
 Prototype Design Pattern, 92  
 Proxy Design Pattern, 97  
 Strategy Design Pattern, 111–113  
 Visitor Design Pattern, 125

**XMLHttpRequest, 97**

**XPath, 68**

## Y

**Yahoo!, 56**

**Your Contacts, 201**

## Z

**ZCE. See Zend Certified Engineering**

**Zend Certified Engineering (ZCE), 9**

**Zend Framework, 6, 11**

Adapter Design Pattern, 14  
 Design Patterns, 13  
 Factory Design Pattern, 13–14  
 Iterator Design Pattern, 14  
 Observer Design Pattern, 14–15  
 Singleton Design Pattern, 13

**Zend Optimizer, 13**

**Zend Studio, 13**

**.zip, 50**