

# Index

## SYMBOLS

`$(document).read()`, **jQuery**, 143

## A

**abstraction, of data layer**, 93–94

**acceptable verbs**, 38

**[AcceptVerbs] attribute**

overview of, 38

testing existence of Register action and correct signature, 39–40

**Accordion UI control**, 251

**account management, high-level design for**, 29–30

**AccountController class**

creating tests related to membership actions, 37

testing user registration, 41–42

**AccountControllerTest class**, 39

**action filters, ASP.NET MVC**, 9

**Action method**

image hosting, 208

message composition, 173

message retrieval, 180–181

**ActionFilterAttribute, creating action filters**, 9

**ActionResult, testing validity of**, 37

**ActiveX**, 187

**Add method**

adding messages to repository, 171

assigning unique ID, 97–98

client and server validation testing and, 77–80

composing messages, 170–171

handling validation of data layer repository, 94–96

`InMemoryContactService` and, 121–122

usage tracking and, 241–242, 246, 248

validation testing and, 80

**advice, AOP**, 13

**AJAX**

displaying images after upload, 206

message templating and, 219–220, 224

POST vs. GET, 258

submitting form data with, 201–204

**Alert boxes**, 196

**Amazon EC2 (Elastic Computer Cloud)**, 260

**AOP (aspect-oriented programming)**, 12–13

**AppHelper class**

creating for membership test, 46–48

testing validity for contact import, 155–156

**AppHelperTest class**

creating for membership test, 46

validating email address for registration, 45–47

**.ascx user controls**, 132–134

**aspect, AOP**, 13

**aspect-oriented programming (AOP)**, 12–13

## **ASP.NET MVC framework, overview, 6–10**

- action filters, 9
- controller, 7–8
- HTML helpers, 9
- model, 7–8
- ModelState, 10
- overview of, 6
- routing, 8
- TempData, 9–10
- view, 7–8
- ViewData, 8–9
- WebForms model vs., 6–7

## **AssertCreateValidationError helper function, for asserting validation error, 169**

### **assertions**

- refactoring code and, 67–72
- refactoring design and, 66
- similarities pointing to need for refactoring, 72

## **Assert.Throws, in exception testing, 79**

## **attributes, validation, 84**

## **authentication, forms, 54–55**

## **Authorize action filter, 9**

## **Authorize attribute, 168**

## **Autofac, IoC container, 21**

## **B**

## **banners, as images, 195**

## **billing and subscriptions**

- constructing payment service, 226–228
- design, 225–226
- overview of, 225
- PayPal implementation, 228–230
- problem statement, 225
- summary, 236
- View of, 233–236

## **BlackTie theme, ThemeRoller, 255**

### **Browse actions/methods**

- browsing messages, 179–180
- contact management and, 129
- IContactService interface, 127
- message composition and, 178
- redirection to following import, 149
- sorting and, 134

### **browsing contacts**

- adding pager as user control, 132–134
- controller action for, 127
- design, 111
- mapping URL to Contact controller, 129
- overview of, 125
- PagedList class and, 127–129
- populating repository with test data, 126–127
- sorting and, 134–136
- testing contract retrieval for logged-in user, 129–130
- testing retrieval of one page of contacts, 126
- view for iterating through contact list, 130–132

### **bugs**

- creating contacts and, 121, 125
- importing contacts and, 161–165
- refactoring not fixing, 259
- TDD (Test Driven Development) and, 11

## **business logic object, 77**

### **buttons**

- creating for PayPal, 234
- CSS interaction state classes applied to, 256
- as images, 195
- Insert Image button, 196–198

**C****caching**

- images, 257–258
- output caching, 258

**callbacks**

- payment processing and, 225
- verifying performance by callback method, 226–228

**campaign report, reports and stats design, 5****Cascading Style Sheets (CSS)**

- in jQuery UI Library, 252–256
- minification of, 257

**classes**

- ASP.NET MVC model, 7
- defining user class, 122

**click events, images, 209****client-side validation, 87–92**

- benefits of, 73
- options for, 87–88
- test class for, 88–92
- validating contact view, 115–116

**cloud computing, 260****code coverage, 19–20****code refactoring. See refactoring code****ComplexModelBinder, 69****composing messages**

- adding instance of `IMessageService`, 170
- `Authorize` attribute for controlling access to message creation, 168
- coding edit functionality, 180–181
- creating message controller, 168
- design process, 167
- helper methods for populating repository, 176–177
- listing existing messages, 174–175
- overview of, 167

- populating repository with messages, 179–180
- problem statement, 167
- Request Validation and, 185
- returning error message if requested message does not exist, 181–182
- saving changes to repository, 183
- summary, 185–186
- testing if user can see messages, 182
- testing that message is added to repository, 171–172
- validating message name, 168–169
- views of, 172–173, 178–179, 183–184

**compression, gzip for, 257****constructor parameters, dependency injection and, 103, 106–107****Contact Lists, 110****contact management**

- browsing contacts. See browsing contacts
- checking validity of contact information (email, date of birth), 114–116
- contact defined, 111
- controller for, 112–113
- `Create` action, 113–114, 119
- creating contacts, 111
- deleting contacts, 111, 143–144
- design, 110–111
- editing contacts. See editing contacts
- establishing relationship between user and contact, 121–122
- high-level design for, 32–33
- importing contacts. See importing contacts
- instantiating controller using dependency injection, 123–125
- notification of successful creation of contact, 119–120
- overview of, 3–5, 109

### **contact management (*continued*)**

- problem statement, 109–110
- summary, 145
- testing contact creation, 116–118
- testing invalid email, 118
- testing that multiple users can access same contact, 123
- testing uniqueness of email addresses, 120–121
- TinyMCE integration with contact management systems, 188

### **ContactController**

- creating contacts and, 112–113
- importing contacts and, 148

### **ContactServiceTests class, 159**

### **containers, IoC. See IoC (Inversion of Control)**

### **controllers**

- `AccountController` class, 37, 39, 41–42
- ASP.NET MVC, 7–8
- Browse controller action, 127–130
- configuring for dependency injection, 106
- contact editing, 137
- contact management, 112–113
- `ContactController` class, 148
- creating message controller, 168
- `GalleryController` class, 199–200
- helper method for instantiating, 177
- IoC (Inversion of Control), 123–124
- `MessageController` class, 75, 77, 84–85, 170
- `PayController` controller, 226
- refactoring controller actions, 65
- `SetUp` method and, 176
- template controllers, 213–214
- `TrackerController` controller, 239, 241
- usage tracking, 238–241

### **controls, in jQuery UI Library, 251**

### **Convention over Configuration principle, 17–18**

### **copying/pasting contacts, 147**

### **coupling, benefits of loosely coupled systems, 103**

### **Create action/methods**

- action filters, 9
- contact management, 113–114
- contacts, 3–4
- CRUD operations, 31
- message controller, 168
- messages, 2, 31, 76
- preventing unauthorized access to, 168

### **CreateUser method**

- handling registration failures from Membership Provider, 49–50
- simulating call to membership provider, 40–42

### **creating contacts**

- checking validity of contact information (email, date of birth), 114–116
- controller for, 112–113
- `Create` action, 113–114, 119
- design, 111
- establishing relationship between user and contact, 121–122
- instantiating controller using dependency injection, 123–125
- notification of successful contact creation, 119–120
- testing contact creation, 116–118
- testing invalid email, 118
- testing that multiple users can access same contact, 123
- testing uniqueness of email addresses, 120–121

**cross-cutting concerns, AOP, 12****CRUD (create, read, update, and delete)**

contact management using, 31

`IRepository` pattern and, 94

**CSS (Cascading Style Sheets)**

in jQuery UI Library, 252–256

minification of, 257

**D****data layer**

changing `InMemoryMessageService` to use repository, 95–96

creating `IMessageRepository`, 94–95

design, 94

implementing `IMessageRepository`, 96, 100–101

in-memory repository and, 102

overview of, 93

problem statement, 93

testing ability to save existing messages to repository, 99–100

testing delete functionality, 98–99

testing unique ID assignment, 97–98

**data store. See repositories****databases. See repositories****Date Picker, UI control, 251****delete actions**

contact list, 5

contact management, 111, 143–144

contacts, 4

CRUD operations, 31

message repository, 98–99

messages, 3

**Delete method, verifying removal of messages, 98–99****dependencies, 103. See also DI (Dependency Injection)****Dependency Injection. See DI (Dependency Injection)****design, high-level**

account management, 29–30

contact management, 32–33

environment requirements, 34

membership system, 25–29

message management, 31

overview of, 25

problem, 25

**DI (Dependency Injection)**

design, 104–105

message templating and, 216

Ninject IoC Containers for, 105–108

overview of, 103

PayPal service, 235

principle, 16

problem statement, 103–104

summary, 108

**div element**

for hidden layers, 197

holding images in, 207

**document object model (DOM), 143****DOM (document object model), 143****DRY (don't repeat yourself)**

principle, 16

refactoring code and, 65

**duplicate action, messages, 3****DynamicImage action**

testing file content and filename, 239–241

tracking email message view, 244

**E****EC2 (Elastic Computer Cloud), 260****edge cases, finding/handling, 186****edit actions**

contacts, 4, 137

form submission, 140

messages, 2, 182–183

post, 183

routing and, 8

### **Edit functionality, coding, 180**

### **Edit link, creating, 174**

### **editing contacts**

- adding error message for non-existent contacts, 138–139
- adding `Get` method to `IContactService`, 137
- design, 111
- enforcing editing only by logged-in user, 139–140
- handling form submission, 140–143
- overview of, 136
- testing retrieval of correct contact item, 137

### **Elastic Computer Cloud (EC2), 260**

### **email**

- account management design, 29–30
- adding contact to email address list, 33
- checking missing addresses, 43–44
- checking validity of contact information, 114–116
- components of email messages, 167
- testing email validation, 90–91
- testing invalid addresses, 45–47
- testing invalid email, 118
- testing uniqueness of email addresses, 120–121
- tracking email messages, 244
- uploading images, 195

### **error messages**

- customizing, 139
- displaying, 138–139
- displaying valid, 56
- membership registration and, 42–43, 45, 51
- Request Validation and, 185
- requested messages and, 182
- returning appropriate, 181–182
- validation and, 59

### **ErrorCodeToString method, 51–52**

### **EvenContact web application**

- defined, 1
- reasons for creating, 22
- requirements. See requirements

### **Excel spreadsheets, importing, 147**

### **exceptions**

- `Assert.Throws`, 79
- creating validation framework, 77–84
- helper functions for asserting validation errors, 169
- `Message` class, 170
- `ValidationException`, 78

### **ExpectedException attribute, 78**

## **F**

### **facicon.ico, 257**

### **fields**

- naming conventions, 57
- validating, 59–60

### **file content, testing in usage tracking, 239–241**

### **file upload**

- displaying images after, 206
- image galleries and, 198
- modal JavaScript dialog box for, 201
- testing functionality of, 204–206

### **filenames, testing in usage tracking, 239–241**

### **FileResult, in image hosting, 208**

### **Firefox Bookmarklet, ThemeRoller, 252, 254**

### **Flash, HTML editors and, 187**

### **flash messages**

- flash message pattern, 119–120
- Import action and, 149
- Ruby on Rails, 119

**forms**

- authentication, 54–55
- creating Edit action for form
  - submission, 140
- submitting form data with AJAX, 201
- WebForms suited for Visual Studio, 7

**FormsAuthenticationWrapper class, 54–55**

**front-end optimization tips, 257–258**

**G****galleries**

- browsing image gallery, 195
- interface for image gallery, 198
- template gallery, 216

**GalleryController class, 199–200**

**Get method**

- added to `IContactService` interface, 137
- composing messages and, 181–182
- contact management and, 140
- image hosting and, 207
- message templating and, 215, 221–222
- `MessageService` class, 101–102

**GetPage method, 134, 178**

**global address book, contacts stored in, 3–4**

**GoGrid cloud solution, 260**

**Goode, Troy, 129**

**gzip, for compression, 257**

**H**

**HandleError attribute, 139**

**handshakes**

- creating for PayPal, 228–229
- payment processing and, 225
- verifying performance by callback method, 226–228

**helper functions**

- for asserting validation errors, 169
- HTML helpers, 9
  - `Html.AttributeEncode` helper method, 57
- for instantiating controller, 177
- for populating test repository, 176
  - `ValidationMessage` helper method, 59

**high-level design**

- account management, 29–30
- contact management, 32–33
- environment requirements, 34
- membership system, 25–29
- message management, 31
- overview of, 25
- problem, 25

**homepage, redirecting user to, 38–39**

**horizontal scaling, 259**

**HTML**

- components of email messages, 167
- helpers, 9
- message content, 74
- messages and images, 195
- `Prefix`, 70
- TEXTAREA fields, 188

**HTML editors, 192–193**

- design, 187
- factors in choosing, 188
- jWYSIWYG, 192–193
- NicEdit, 190–191
- problem statement, 187
- TinyMCE, 188
- WYMeditor, 189–190

**Html.AttributeEncode helper method, 57**

**hyperlinks**

- adding to master page, 197
- ease of adding with HTML editors, 187

## icons, in jQuery UI Library, 251

**IContactRepository**, 121

**IContactService**

adding `Browse` method to, 127

creating, 117–118

`GetPage` method, 134

**IContactsImporter**, 151

**IDE Visual Studio 2008**, 7

**IEnumerable**, 127–129

**if statements**

server validation with, 73

writing, 151

**IFormsAuthentication**, 54–55

**IGalleryRepository**, 200–201

**IGalleryService**, 200

**image editors**, 239

**image galleries**

browsing, 195

interface for, 198

overview of, 196

**image hosting**

design, 195–196

displaying images after upload, 206

file upload added to modal JavaScript dialog, 201

file upload and storage in memory, 198–201

gallery interface for, 198

Insert Image button, 196

modal JavaScript dialog box for, 196–198

overview of, 195

problem statement, 195

selecting images and inserting into message body, 209

submitting form data, 202–204

summary, 210

testing file upload functionality, 204–206

testing `FileResult`, 208

testing image retrieval, 206–207

**images**

embedding inline, 187

making small and cacheable for optimization, 257–258

**IMessageAuditService**, 240–241

**IMessageRepository**

creating interface for, 94–95

helper methods for populating, 176–177

implementing, 96

overview of, 171

populating repository with messages, 179–180

saving changes to, 183

testing assignment of unique ID to messages, 97–98

**IMessageRepository**

full implementation of, 100–101

testing ability to save existing messages to repository, 99–100

testing delete functionality of, 98–99

**IMessageService**

changing `InMemoryMessageService` to use repository, 95–96

LINQ lazy loading and, 101

message composition and, 170–172

**IModelBinder**, 69

**import actions and methods**

bugs, 161

calling `Import` method on contacts service, 157–158

flash messages and, 149–150

populating contact database, 151

validating, 160

**importing contacts**

bugs, 161–165

controller for, 148  
 design, 147–148  
 implementation, 159–161  
 instantiating importer instances, 151  
 overview of, 147  
 parser factory, 151–152  
 problem statement, 147  
 string parser, 154–156  
 summary, 166  
 testing, 165–166  
 testing parser factory, 156–159  
 testing success of Import action, 149–150  
 testing that string or uploaded file posted to server, 148–149  
 text file parser, 153–154  
 View for, 150

#### **inline content editors**

jWYSIWYG, 192  
 NicEdit, 190

#### **in-memory. See also IRespository pattern, 121**

contact service, 121–122  
 data layer and, 102  
 message service, 95–96, 100  
 tight coupling and, 104

#### **InMemoryContactRepository, 121**

#### **InMemoryContactService, 121–122**

#### **InMemoryMessageRepository, 100**

#### **InMemoryMessageService**

adapting to use `IMessageRepository`, 95–96  
 problem of tight coupling, 104

#### **Insert Image button**

default dialog, 196  
 executing custom JavaScript, 196–198

#### **Internet, validating all data received via, 73**

#### **Inversion of Control. See IoC (Inversion of Control)**

#### **IoC (Inversion of Control)**

addressing tight coupling with IoC Containers, 104–105  
 choosing between IoC Containers, 21, 105  
 creating controllers and, 123–124  
 overview of, 103  
 principle, 16  
 uniqueness check for logged-in users, 120–122  
 working with Ninject IoC Container, 105–108

#### **IPaymentService, 227**

#### **IQueryable**

Get method and, 101–102  
 PagedList collection class, 127–129

#### **IRespository pattern, 93–94, 102. See also data layer**

#### **ISubscriptionPlanRespository, 231**

#### **IsValidEmail method, 45–47**

#### **IsValidUsername method, 47–49**

#### **ITemplateRespository, 215**

#### **ITemplateService, 214, 217**

## **J**

#### **Java, HTML editors and, 187**

#### **JavaScript**

HTML editors and, 187  
 libraries, 19  
 minification of, 257  
 modal JavaScript dialog box, 196–198, 201  
 using third-party library for client-side validation, 87–88

#### **JavaScript Object Notation (JSON)**

message templating and, 219–220, 224  
 overview of, 203

#### **JavaScript-based editors, 187**

adding template to, 218–219

### JavaScript-based editors (*continued*)

design, 187  
factors in choosing, 188  
jWYSIWYG, 192–193  
NicEdit, 190–191  
problem statement, 187  
saving content as HTML template, 223–224  
TinyMCE, 188  
WYMeditor, 189–190

### joinpoint, AOP, 13

### jQuery

`$(document).read()`, 143  
JavaScript libraries, 20  
jWYSIWYG using, 192  
plug-ins enabling AJAX uploads, 204  
UI Library, 196–197  
UI library, 250–252  
validating contact view, 115–116  
Validation plug-in, 87–90  
WYMeditor using, 190

### JSON (JavaScript Object Notation)

message templating and, 219–220, 224  
overview of, 203

### jWYSIWYG, 192–193

## K

**keep it simple, stupid (KISS), 16**

**KISS (keep it simple, stupid), 16**

## L

**lazy instantiation pattern, 14–15**

**lazy loading, LINQ, 101–102**

### libraries

JavaScript libraries, 19–20  
jQuery UI Library, 196–197, 250–256  
LINQ Dynamic Query Library, 135  
mocking, 22

using third-party library for client-side validation, 87–88

YUI (Yahoo! User Interface Library), 193

**links. See hyperlinks**

### LINQ

lazy loading, 101–102  
sorting strings and, 134–135

**LINQ Dynamic Query Library, 135**

**Liskov substitution principle, 17**

**List method, templates, 221–222**

### lists

adding contact to, 5, 32–33  
Contact Lists, 110  
creating contact list, 4  
deleting contact list, 5  
of existing messages, 174–175  
of images, 209  
iterating through contact list, 130–132  
of messages, 2  
`PagedList` class and, 127–130  
parsing contact list, 151–157  
of system templates, 213

**logged-in users, uniqueness check for, 122**

**logging, using aspect-oriented programming, 12–13**

### login process

designing membership system, 27–28  
refactoring code and, 68

**LoginModel class, 68**

**Lupetti, Antonio, 132**

## M

### master page

adding links to, 197  
creating view, 56–58

### MbUnit

ReSharper test runner and, 34

testing email validation, 46–47

unit testing framework, 17–18

### **membership**

error returned if email is invalid, 45–47

error returned if email is missing, 43–44

error returned if username is invalid, 47–52

error returned if username is missing,  
42–45

forms authentication, 54–55

login process design, 27–28

overview of, 35–36

password confirmation design, 27

problem, 35

redirects to home index on successful test,  
40–42, 52–53

Registration page view, 56–63

registration view test, 37–38

reset password process, 28–29

test classes for, 36

testing user registration, 39–40

user registration design, 25–26

### **Membership Provider**

creating user with, 40

handling registration failures from, 49–52

simulating call to, 40–42

### **Message class, 170**

#### **message ID**

saving to database, 246

testing for, 243

### **message queue, 31**

#### **message templating**

creating template controller, 213–214

design, 211–213

enabling users to save own templates,  
219–224

getting list of system templates, 213

overview of, 2, 211

problem statement, 211

summary, 224

updating WYSIWYG editor for, 218–219

View, 216–217

### **MessageAudit**

creating model for, 240

testing redirection to URL, 245–247

tracking type of action performed by user,  
243–244

### **MessageController**

creating, 75, 170

test classes, 74–75

validation testing, 84–85

### **messages**

adding page to all existing messages, 174

assigning unique ID to, 97–98

components of, 74

composing. See composing messages

design requirements, 2–3

high-level design for, 31

saving changes to existing, 99–100

selecting image and inserting into message  
body, 209

sending to users. See contact management

testing ability to save existing messages to  
repository, 99–100

testing assignment of unique ID to, 97–98

verifying removal of, 98–99

### **MessageService class, 81, 101–102**

### **methods, naming test, 36**

### **Microsoft Windows Azure, 260**

### **minification, of JavaScript and CSS, 257**

### **mocking**

DI (Dependency Injection) combined  
with, 103

### mocking (continued)

- Files collection, 199
- forms authentication, 54–55
- IContactsImporter, 151
- IMessageRepository, 95
- libraries, 22
- Mocks class, 67–68
- payment processing, 230–233
- payment service, 227
- refactoring design and, 66
- refactoring mocks, 67–68
- setting up mock objects for message composition, 177
- similarities pointing to need for refactoring, 72
- simulating call to membership provider, 40–42
- WebRequest creation, 229–230

**modal JavaScript dialog box, 196–198, 201**

**model binders, in refactoring, 66, 69–70**

**model-based validation, 85**

**ModelState, ASP.NET MVC, 10**

**Model-View-Controller. See MVC (Model-View-Controller)**

**Moq, 22**

**MSDN LINQ Dynamic Query Library, 135**

**MSTest, unit testing framework, 17**

**Muehsig, Robert, 132**

**MVC (Model-View-Controller), 6**

**MVC (Model-View-Controller), in ASP.NET, 6–10**

- action filters, 9
- controller, 7–8
- HTML helpers, 9
- model, 7–8
- ModelState, 10
- overview of, 6

- routing, 8
- TempData, 9–10
- view, 7–8
- ViewData, 8–9
- WebForms model vs., 6–7

## N

### naming conventions

- form fields, 57
- test classes and test methods, 36
- tests related to actions, 37
- views, 56

### NCover, 19

### NicEdit, 190–191

### Ninject ([www.ninject.org](http://www.ninject.org))

- dependency injection with, 105–108
- IService interface and, 118
- IoC Containers and, 105
- message templates and, 216
- PayPal service and, 235
- uniqueness check for logged-in users, 120–122

### notification messages

- notifying user contact message successfully created, 119
- showing save successfully created, 173

### notification URL, payment processing and, 225

### null object pattern, 14

## O

### objects

- business logic object, 77
- indirection of object instantiation, 103–104
- null object pattern, 14
- Template objects, 213

**open-closed principle, 15**

**optimizing code**

- design, 250
- front-end optimization tips, 257–258
- gzip for compression, 257
- minification of JavaScript and CSS, 257
- output caching, 258
- overview of, 256
- refactoring code, 258–259

**Outlook Personal Files, importing, 147**

**output caching, 258**

## P

**page, adding to existing messages, 174**

**PagedList collection class, 127–130**

**pager user control, 132–134**

**pagination, of list of images, 209**

**Pagination class, 129**

**Pagination.ascx user control, 132–134**

**parsers**

- creating parser factory, 151–153
- string parser, 154–156
- testing parser factory, 156–159
- text file parser, 153–154

**password**

- confirmation, 27
- reset process, 28–29
- validating, 44

**patterns, design**

- flash message pattern, 119–120
- `IRepository` pattern. See `IRepository` pattern
- lazy instantiation, 14–15
- null object, 14
- repository, 15
- strategy, 13–14

**Patterns of Enterprise Application**

**Architecture (Fowler), 13**

**PayController controller, 226**

**payment service, 226–228. See also billing and subscriptions**

**PayPal**

- account management design, 29–30
- implementing PayPal service, 228–230
- payment processing with, 230–233
- `View`, 228–230

**performance, improving perceived, 257**

**PerformHandshake, 227**

**Plain Old CLR object (POCO) classes, 7**

**plaintext messages, 74**

**POCO (Plain Old CLR object) classes, 7**

**PopulateRepository method, 126**

**pop-ups**

- jWYSIWYG and, 192
- NicEdit and, 191
- TinyMCE and, 188
- WYMeditor and, 190

**Prefix, HTML, 70**

**principles**

- Convention over Configuration, 17–18
- defined, 15
- Inversion of Control/Dependency Injection, 16
- KISS, 16
- open-closed, 15
- overview of, 16
- single responsibility, 16–17
- YAGNI, 15

**ProcessPayment method, 232–233**

**promotional badge, as images, 195**

**public properties, injecting dependencies with, 103**

## Q

**queue-monitoring service, 31**

## R

**RAD (Rapid Application Development), 7**

**range attribute, validation attributes, 84**

**Rapid Application Development (RAD), 7**

**read action, CRUD operations, 31**

**RedirectToRouteResult value, 39**

**Redmond theme, 254**

**refactoring code, 65–72**

assertions and, 70–72

design, 66, 250

generic assertions and, 66–67

mocking and, 67–68

model binders and, 69–70

optimizing code and, 258–259

overview of, 65

PayPal implementation, 229

problem statement, 65–66

setup method running for every test, 67

summary, 72

**Register action tests**

error returned if email invalid, 45–47

error returned if email missing, 43–44

error returned if username invalid, 47–52

error returned if username missing, 42–43

getting to register view, 37–38

redirects to home index on success,  
40–42, 52–53

successfully registration of new  
user, 39–40

**registration (signup) process**

action tests. See Register action tests

forms authentication, 54–55

login process, 27–28

overview of, 35–36

password confirmation, 27

problem, 35

reset password process, 28–29

user registration, 25–26

**regular expression attribute, validation  
attributes, 84**

**reports and stats, viewing, 5**

**repositories**

adding contacts to, 159–161

adding message to, 171

creating for usage tracking, 240–241

helper function for populating test  
repository, 176

`IContactRepository`, 121

`IGalleryRepository`, 200–201

`IMessageRepository`. See

`IMessageRepository`

`IRepository` pattern as alternative to  
testing, 93

for PayPal service, 235

`PopulateRepository` method, 126–127

populating with messages, 179

repository pattern, 15

saving changes to, 183

**Request Validation, 185**

**Request.Files, 148, 198**

**Request.Url property, 209**

**required attribute, validation attributes, 84**  
**requirements**

aspect-oriented programming (AOP), 12–13

code coverage, 19–20

contact management design, 3–5

design, 1–2

IoC containers, 21

JavaScript libraries, 19

message management design, 2–3

miscellaneous design, 5–6  
 mocking libraries, 22  
 Model-View-Controller (MVC), 5–10  
 patterns, 13–15  
 principles, 15–18  
 problem, 1  
 reports and stats design, 5  
 Test Driven Development (TDD), 10–11  
 tools and frameworks, 18–22  
 unit testing framework, 17–18

### **ReSharper**

MbUnit working with, 17–18  
 overview of, 34

### **rich text capability, 193**

### **Rich Text Editor, YUI, 20, 193**

### **routing, 8**

### **Row Tests, MbUnit, 17**

### **Ruby on Rails, 119**

## **S**

### **save actions/methods**

message templates, 219–224  
 messages, 3  
 notification of successful save, 173  
 Save Message button, 184  
 saving changes to repository, 183  
 testing ability to save existing messages to repository, 99–100  
 testing presence of email address before saving data to database, 242–243  
 usage tracking and, 246

### **scaling applications**

clouds and, 260  
 design, 250  
 scale out (horizontal scaling), 259  
 scale up (vertical scaling), 259

### **scripting attacks, 185**

### **search actions**

contacts, 4  
 messages, 3

### **security check, enforcing in service layer, 140**

### **send action, messages, 3, 31**

### **server-side validation**

benefits of, 73  
 creating test class for, 74–77  
 creating validation framework, 77–84

### **SetAuthCookie method, forms**

#### **authentication, 54–55**

### **SetFakeStatus method, handling**

#### **registration failures, 49–50**

### **SetUp method**

initiating controller, 176  
 refactoring code and, 67

### **Shared folders, views, 56**

### **single responsibility principle**

Liskov substitution, 17  
 overview of, 16–17

### **singleton, using for in-memory data class, 125**

### **software principles, 15–18**

### **sorting**

applying strategy pattern to, 14  
 arguments required for, 134  
 Browse actions and, 134–136

### **startup methods, refactoring code and, 67–68**

### **stats, viewing, 5**

### **strategy pattern, 13–14**

### **styles, in jQuery UI Library, 251**

### **subscriptions. See also billing and subscriptions**

account management design and, 29–30

## **subscriptions (continued)**

- Subscribe Now button, 234
- subscribing to PayPal, 231–232
- SubscriptionPlan model, 236
- unsubscribing from PayPal, 232–233, 235

## **T**

### **tables, 93**

### **Tabs, UI control, 251**

### **TDD (Test Driven Development)**

- code coverage in, 19–20
- MVC suited for, 6
- overview of, 10–11
- slowness as factor in, 93
- testing/fixing bugs, 161
- unit testing framework, 17–18
- validation testing, 77

### **TempData**

- adding error message to, 138–139
- ASP.NET MVC and, 9–10
- creating flash messages, 119–120
- overview of, 119

### **Template objects, 213**

### **templates. See message templating**

### **test classes**

- client-side validation, 88–92
- contact management, 112
- importing contacts, 151
- membership, 36
- message composition, 168, 175
- MessageControllerTest, 74–75
- naming conventions, 36
- server-side validation, 74–77

### **Test Driven Development. See TDD (Test Driven Development)**

### **test runners, 34**

### **Test Utility, YUI, 20**

### **TestDriven.net, 34**

### **text**

- components of email messages, 167
- importing text files, 147
- parsing text files, 151–157
- styling, 187

### **Text property, 171**

### **TEXTAREA fields, HTML, 188**

### **ThemeRoller, jQuery, 250–254**

- CSS Framework, 252–256
- Firefox Bookmarklet, 252, 254
- overview of, 250

### **thumbnails, 209**

### **TinyMCE, 188**

### **TrackerController controller, 239, 241**

## **U**

### **UI (user interface)**

- CSS Framework in jQuery UI Library, 252–256
- design, 249–250
- jQuery UI Library, 250–252
- ThemeRoller, 252, 254
- WYSIWYG Editor, 195

### **unique IDs, messages, 97–98**

### **unit testing framework**

- overview of, 17–18
- Register action tests. See Register action tests

### **update action, CRUD operations, 31**

### **Upload action, GalleryController class, 199–200**

### **Url.Action ("Register") helper method, 57**

### **URLs**

- inserting URL to a picture, 196
- mapping URL to Contact controller, 129

refactoring code and, 71

`Request.Url` property, 209

### usage tracking

creating controllers for, 238–241

design, 237–238

problem statement, 237

testing presence of email address before

    saving data to database, 242–243

testing redirection to URL, 245–247

testing type of action being audited,  
247–248

tracking type of action performed by user,  
243–244

### user controls, pager, 132–134

### user interface. *See* UI (user interface)

### user registration, designing membership system, 25–26

### user report, 5

### usernames

checking validity of, 47–52

filtering messages by, 182

### users, 243–244

account management, 29–30

defining user class, 122

designing membership system, 25–29

relationship between user, contact, and  
Contact List, 110

## V

### validation

adding to contact view, 114–115

attributes, 84

client-side. *See* client-side validation

of data layer repository, 94–96

design, 74

email addresses, 43–44

of fields, 59–60

file upload and, 206

helper functions, 169

of import actions and methods, 160

of `MessageController`, 84–85

model-based, 85

moving out of message controller, 77

overview of, 73

problem statement, 73–74

Request Validation, 185

server-side. *See* server-side validation

summary, 92

testing all fields, 79–80

usernames, 47–52

### `ValidationException`, 78, 80–81

### `ValidationMessage` helper method, 59

### verbs, acceptable, 38

### vertical scaling, 259

### View

ASP.NET MVC and, 7–8

for billing and subscriptions, 233–236

for composing messages, 172–173,  
178–179, 183

for contact management, 115–116, 130

conventions for, 56

for creating contacts, 112–114

for editing contacts, 137

for file upload, 207

for importing contacts, 150

for iterating through contact list, 130–132

for message templating, 216–217

for My Templates, 222–223

for PayPal, 228–230

of Registration page, 56–63

sending data to, 8–9

for title setup, 148

## View state

---

### **View state, master page, 60–63**

#### **ViewData**

ASP.NET MVC, 8–9

ModelState property, 10

TempData vs., 9–10

testing for template item, 217

testing `mytemplates` data, 222–223

### **viewing, compared with editing, 4**

#### **ViewPage, ASP.NET MVC, 7**

#### **Views folder, 56**

#### **ViewUserControl, 7**

### **virtual machines, cloud solutions and, 260**

#### **Visual Studio 2008, 7**

## W

### **W3C XHTML specification, 189**

#### **Web references**

free test runner, 34

ReSharper, 34

#### **WebForms model, ASP.NET**

defined, 6

strengths of, 7

#### **WebRequest, mocking creation of, 229–230**

#### **Windows Azure, Microsoft, 260**

### **WYMeditor, 189–190**

#### **WYSIWYG (what you see is what you get) editors**

adding template to, 218–219

design, 187

factors in choosing, 188

jWYSIWYG, 192–193

NicEdit, 190–191

problem statement, 187

saving content as HTML template, 223–224

TinyMCE, 188

WYMeditor, 189–190

#### **WYSIWYM (what you see is what you mean) editors, 189**

## X

### **XHTML, 189**

## Y

### **YAGNI (you ain't gonna need it) principle, 15**

#### **YUI (Yahoo! User Interface Library)**

Rich Text Editor and, 193

using, 20









