

Index

SYMBOLS

<% %> blocks

- <%= %> blocks vs., 97
- in creating a map Partial view, 129
- Views and, 253

A

AAA testing pattern, 149

abstract base classes

- ActionResult, 233–235
- ActionSelectorAttribute, 242
- ContentResult, 235
- ControllerBase, 229
- EmptyResult, 235
- FileResult, 236
- HttpContextBase, 359
- HttpRequestBase, 359, 419–420
- JavaScriptResult, 237
- JsonResult, 236–237
- PartialViewResult, 238
- RedirectResult, 237
- RedirectToRouteResult, 237
- RoutablePage, 224
- ViewResult, 237

abuse

- AJAX and, 279–280
- IQueryable<T> and, 110

AcceptVerbsAttribute

- applying, 242–243
- HTTP-POST and, 63

AccountController

- [Authorize] filter and, 114–115
- convention-based naming and, 54
- forms authentication and, 111–114

action filter(s)

- custom, types of, 313–315. *See also custom filters*
- function of, 311–312
- HttpRequest validation, 342–343
- OutputCacheAttribute as, 308–310

action methods

- [Authorize] to secure, 346
- ActionResult, 238–239
- adding to DinnersController class, 36–37
- completing edit implementations, 71
- Controllers and. *See Controller class and actions*

- HTTP-GET, 57–61, 72–74, 78–80
- HTTP-POST, 62–66, 74–78, 80–82
- implicit ActionResult, 239–240
- JSON-based AJAX Search, 140–145
- mapping with ControllerActionInvoker, 241–243
- passing data to. *See Model Binders*
- redirecting, 360–361
- Register, 122–123

ActionExecutedContext, 314

ActionExecutingContext, 313

ActionFilterAttribute, 313

ActionNameAttribute, 241–242

ActionResult

- helper action methods, 238–239
- implicit action methods, 239–240
- new View engine or, 275
- overview, 233–234
- types, 235–238

ActionSelectorAttribute, 242

Active Injection, in cross-site scripting, 334, 336–338

Active Server Pages (ASP)

- development of, 177
- MVC and, 167

AddRuleViolations helper method, 70–71

advanced routing, 217–218. *See also routing; routing, introduction*

Agile software

- BDUF vs, 368–369
- testability, 369–370, 371

AJAX

- advantages of, 278
- disadvantages of, 278–280
- examples. *See AJAX examples*
- overview, 277–278
- summary, 303

AJAX, enabling RSVPs

- adding a jQuery animation, 125–127
- calling Register, 123–125
- implementing Register, 122–123
- indicating if RSVP'd, 120–122
- overview, 119

AJAX examples

- auto-complete textbox, 292
- filtering data with a Select box, 295–296
- handling disabled scripting, 280–284
- implementing auto-complete, 292–295
- modal popup code, 297–299

AJAX examples (continued)

AJAX examples (continued)

- modal popup with jQuery, 296–297
- overview, 280
- rating control, 299–302
- things you may not know, 288–290
- updating an HTML element, 290–291
- using `Partials` and, 284–287

AJAX map, integrating

- calling a JSON-based AJAX method, 141–145
- creating a `Map.js` utility library, 129–130
- creating a `Partial` view, 127–129
- implementing a JSON-based AJAX method, 140–141
- implementing location search, 136–140
- integrating with `Create` and `Edit` forms, 131–134
- integrating with `Details` view, 134–136
- overview, 127

alternative View Engines. See *View engines, alternative*

ambient values, in URL generation, 211–214

angle bracket(s)

- control, Web Forms and, 183
- generator, View engine as, 266–267

animation, jQuery, 125–127

Apache Struts, 169–170

application, testability

- avoiding singletons and static methods, 373–376
- future-proofing with interfaces, 371–373
- overview, 371
- Single Responsibility Principle, 373

applications

- basic Web Forms, 176–179
- creating ASP.NET MVC. See *ASP.NET MVC, application security*. See *security, application*

ASP. See *Active Server Pages (ASP)*

ASP.NET > ASP.NET MVC

- guiding tenants. See *ASP.NET MVC, guiding tenants*
- overview, 175
- reactions to ASP.NET MVC, 187–188
- strengths. See *ASP.NET Web Forms, strengths*
- summary, 196
- where Web Forms doesn't fit, 181–183
- why "(ASP.NET > ASP.NET MVC) == true". See *ASP.NET MVC, application*

ASP.NET MVC

- empty project, creating, 407
- first Web Forms and, 167–168
- routing, understanding of, 37–39
- Test Driven Development with. See *Test Driven Development (TDD)*
- together with ASP.NET Web Forms. See *Web Forms and MVC together*

ASP.NET MVC, application

- convention over configuration, 190–194
- initial procedures, 188–190
- request lifecycle, 196
- third request, 194–195

ASP.NET MVC, filters

- `AuthorizeAttribute`, 306–308
- custom filters. See *custom filters*
- exception filters, 310–311
- `OutputCacheAttribute`, 308–310
- overview, 305

ASP.NET MVC, guiding tenants

- maintainability, approaches to, 185–186
- orchestration vs. composing, 184
- overview, 184
- separation of concerns, 184–185
- testability, caring about, 186–187

ASP.NET MVC, the new kid

- cost/benefit of Web Forms, 172–173
- is this Web Forms 4.0, 171–172
- overview, 170–171
- serving methods, not files, 171
- should you fear it, 173–174
- why not Web Forms?, 172

ASP.NET Session, 403–404

ASP.NET Web Forms

- ASP.NET MVC as an alternative to, 171–172
- basic application, 176–179
- cost/benefit of, 172–173
- differences from ASP.NET MVC, 173–174
- importance of events in, 180–181
- the leak in, 181–182
- together with ASP.NET MVC. See *Web Forms and MVC together*
- using routing with, 222–224
- why not, 172

ASP.NET Web Forms, strengths

- basic Web Forms application, 176–179
- importance of events, 180–181
- overview, 175–176

asynchronous requests, 278

Atwood, Jeff, 343–344

authentication

- authorization vs., 346
- forms, the `AccountController` and, 111–114
- overview, 110
- understanding, 110–111

authorization

- authentication vs., 346
- definition and understanding, 110–111
- of the `/Dinners/Create` URL, 114–115
- overview, 110
- showing/hiding `Edit` and `Delete` links, 118–119
- using the `User.Identity.Name` property, 116–118

authorization filters

- `AuthorizeAttribute` as, 306–308
- function of, 311–312
- writing custom, 317–318

- [Authorize] filters
 - for the /Dinners/Create URL, 114–115
 - securing, 346
- AuthorizeAttribute
 - included with ASP.NET MVC, 306–308
 - routing and, 223
- auto-complete textbox
 - as an AJAX tool, 292
 - implementing with Microsoft ASP.NET AJAX, 292–295

B

- back button, 279
- basic Web Forms application, 176–179
- basic Web Forms applications. *See also* ASP.NET MVC, application
- BDUF (Big Design Up Front)
 - facets of, 368
 - general process, 368–369
 - testability, 369
- benefits, of Web Forms, 172–173
- Big Design Up Front. *See* BDUF (Big Design Up Front)
- blur events, 131
- Boo language, Brail and, 273–274
- bookmarks, AJAX, 279
- botnets, Storm, 332–334
- boundaries, in unit testing, 353
- Brail View engine, 273–274
- brittle condition, 405
- business logic, implementing
 - Dependency Injection with Structure Map, 389–392
 - overview, 383–385
 - services gone wild, 385–386
 - setting Controller dependencies, 386–388
 - using Dependency Injection, 389
- business rule logic, 31–34

C

- Capture the Flag, 329–330
- Castle Project
 - MonoRail and, 170
 - NVelocity and, 272
- catch-all parameter, 206–207
- classes
 - abstract base. *See* abstract base classes
 - Controller. *See* Controllers
 - data model, 22–25, 30–34
 - Dinner model, 84, 147–149
 - DinnerRepository.
 - See* DinnerRepository classes
 - DinnersController.
 - See* DinnersController classes
 - FakeDinnerRepository, 154–157
 - HtmlHelper, 257
 - LINQ to SQL, adding, 21–22
 - NerdDinnerDataContext, 25–26
 - Route, 215–216
 - RSVPController, 122–123
 - ViewModel, custom-shaped, 92
- Classic ASP (1999), and ASP.NET MVC, 187, 253
- client IDs, 183
- ClubSite Starter Kit
 - adding images and styling, 410–411
 - creating an empty ASP.NET MVC project, 407
 - implementing the structure, 415–418
 - migrating to ASP.NET MVC. *See* Web Forms to MVC, migrating
 - MVC Controllers, 413–415
 - overview, 407
 - replacing server controls, 415–418
 - setting up routing, 411–413
 - uploading files/working with images, 418–420
- codes
 - clarifying with partial view templates, 97–98
 - jqModal, 297–299
 - TDD and. *See* Test Driven Development (TDD)
 - testability of. *See* design patterns, testability
- composing, vs. orchestration, 184
- concatenation, string, 364
- confused deputy, 339–341
- constraints, in defining routes, 205–206
- Content directory, 410–411
- content regions, 43
- ContentResult, 235
- context objects, action filter, 313
- control replacement, for migration to MVC, 415–418
- Control Tree, 405
- Controller class and actions
 - action methods, 230–231
 - overview, 229
 - working with multiple parameters, 233
 - working with parameters, 231–232
- ControllerActionInvoker
 - action method selection, 241
 - applying AcceptVerbsAttribute, 242–243
 - applying ActionNameAttribute, 241–242
 - applying ActionSelectorAttribute, 242
 - invoking actions, 244
 - mapping parameters, 243
 - overview, 240
 - passing data to actions. *See* Model Binders
- Controllers
 - ActionInvoker. *See* ControllerActionInvoker
 - ActionResult and. *See* ActionResult
 - actions and. *See* Controller class and actions
 - ClubSite MVC, 413–415
 - ControllerBase class and, 229
 - definition, 225

Controllers (continued)

Controllers (continued)

- function in MVC, 190–194
 - history of, 225–227
 - IController interface and, 227–228
 - in MVC, 166–167
 - sanitizing user input in, 248–249
 - securing, 345–346
 - setting dependencies manually, 386–388
 - summary, 249
 - testing with TDD, 360
- Controllers, NerdDinner application**
- adding a `DinnersController` controller, 35–36
 - adding `Index` and `Details` action methods, 36–37
 - overview, 34–35
 - passing data to view templates, 86–87
 - understanding ASP.NET MVC routing, 37–39
 - using the `DinnerRepository`, 39–40
- convention over configuration, 190–194**
- convention-based naming structure, 54–56**
- cookie theft**
- application security and, 343–344
 - preventing, 344
- cost, of Web Forms, 172–173**
- coupling, and dependency injection, 152**
- Create view templates**
- integrating Map support with, 131–134
 - revisiting, 92–93
- `Create.aspx`
- Dinner form, 92–93
 - partial view templates and, 95–96
- Cross-Page Posting, 404–405**
- cross-site request forgery (CSRF)**
- overview, 339–342
 - preventing, 342–343
- cross-site scripting (XSS)**
- active injection, 336–338
 - overview, 334
 - passive injection, 334–336
 - preventing, 338–339
- CRUD (Create, Update, and Delete) support.**
See Dinners, CRUD support for
- CSRF (cross-site request forgery)**
- overview, 339–342
 - preventing, 342–343
- custom constraints, for advanced routing, 217–218**
- custom filters**
- action type properties, 313–314
 - overview, 311–312
 - writing action, 316–317
 - writing authorization, 317–318
 - writing exception, 319
- custom-shaped ViewModel classes, 92**

D

Dark Dante, 328–329

data access, testable

- creating the model, 377–379
- implementing a Repository stub, 380–381
- implementing the real thing, 382–383
- Northwind Product Repository, 379–380
- overview, 376–377

data model classes

- creating with LINQ to SQL, 22–25
- integrating validation/business rule logic with, 30–34

Database, NerdDinner application

- adding data to tables, 20
- creating a SQL Server Express database, 14–15
- creating tables in, 15–18
- implementing location search in, 136–140
- overview, 13
- setting up a foreign key relationship, 18–20

database, SQL Server Express, 14–15

de Oliveira, Hamilton Verissimo, 170

DEF CON Capture the Flag, 329–330

default unit tests, 353–355

default values

- defining routes, 199
- in URL generation, 202–205

deferred execution, 102

DeJardin, Louis, 274–275

Delete action links, 118–119

deleting

- with the `DinnerRepository` class, 30
- Dinners, 118–119

dependency injection

- frameworks, 158–159
- unit testing and, 152

Dependency Injection (DI)

- implementing business logic and, 389
- with `StructureMap`, 389–392

design patterns, testability

- Agile software development, 369–370
- Big Design Up Front (BDUF), 368–369
- designing your application for.
See application, testability
- implementing business logic. *See business logic, implementing*
- overview, 367
- summary, 392
- testable data access. *See data access, testable*
- testing to prove you're done, 371
- writing testable code, 370–371

Details action methods

- adding to `DinnersController` class, 36–37
- showing/hiding links, 118–119

Details view templates

- implementing, 44–49
- integrating Map support with, 134–136

differences, MVC vs. Web Forms, 173–174**Dinner model classes**

creating tests for, 147–149
 CRUD wrap-up and, 84

DinnerRepository classes

creating, 26–28
 dependency injection and, 152
 from `DinnersController`, 39–40
 extracting an `IDinnerRepository` interface,
 152–154
 retrieving, updating, inserting and deleting
 with, 28–30

Dinners, CRUD support for

`AddRuleViolations` helper method and, 70–71
 complete `Edit` action method implementations, 71
 CRUD wrap-up, 83–86
 handling errors, 66–68
 with `Html.BeginForm` helper method, 61–62
 with `Html.TextBox` helper method, 62
`Html.ValidationMessage` helper method and, 69
`Html.ValidationSummary()` helper method
 and, 70
 implementing the HTTP-GET create action
 method, 72–74
 implementing the HTTP-GET delete action
 method, 78–80
 implementing the HTTP-GET edit action method, 57–61
 implementing the HTTP-POST create action
 method, 74–78
 implementing the HTTP-POST delete action
 method, 80–82
 implementing the HTTP-POST edit action
 method, 62–66
 model binding security, 82–83
 understanding `ModelState` collection, 68–69
 URLs handled by `DinnersController`, 56–57

DinnersController classes

adding a controller, 35–36
 adding index and details action methods to, 36–37
 creating. *See Dinners, CRUD support for*
 creating unit tests for, 150–151
 dependency injection and, 152
`DinnerRepository` from, 39–40
 editing of. *See Dinners, CRUD support for*
`Index` action method recap, 102
 testing `Edit` function of, 159
 updating to support constructor injection, 154
 URLs handled by, 56–57
 using views with, 40–42

`/Dinners/Create` URL, **114–115**

directory structure

naming in ASP.NET MVC, 54–56
 naming in Web Forms, 395–396
`NerdDinner`, examining of, 6–9

disabled scripting, 280–284**disadvantages, of AJAX, 278–280****Django, 169****Domain, implementing, 377–379****DRY (Do Not Repeat Yourself) Principle**

application with master view templates, 98–101
 application with partial view templates, 96–98
 ASP.NET MVC and, 171
 definition of, 92
 Ruby on Rails and, 168

E**Edit action**

links, showing/hiding, 118–119
 unit tests, creating, 159

edit errors

in ASP.NET MVC, 66–68
 HTML helper methods and, 68–69

Edit view templates

integrating Map support with, 131–134
 revisiting, 92–94

Edit.aspx

Dinner form, 92–94
 partial view templates and, 95–96

embedded URL values, 104–106**empty ASP.NET MVC project, 407****EmptyResult, 235****enemy's mind, knowing**

deception and hacking into the server
 room, 330
`DEF CON Capture the Flag`, 329–330
 Kevin Poulsen case, 328–329
 overview, 327–328
 social engineering/ Kevin Mitnick, 330–331

Entity Framework (EF), and AdventureWorks, 379**errors**

handling, in MVC, 66–68
 HTML helper methods and, 68–69
 reporting, application security and, 345

event models, in Web Forms, 173**event-driven programming, 225–226****EventsController**

as a `ClubSite` MVC Controller, 413
 creating an `Index` view for, 417

examples, AJAX. *See AJAX examples***exception filters**

function of, 311–312
`HandleErrorAttribute`, 310–311
 writing custom, 319

[ExpectedException] attribute, 363**extensibility, route, 218–221****extension methods**

`HtmlHelper` class and, 257
 implementing view helpers as, 362–364

FakeDinnerRepository classes

F

FakeDinnerRepository classes

- creating, 154–157
- using with unit tests, 157–159

FileResult, 235

- description of, 236

FileNew Project, NerdDinner application

- examining the directory structure, 6–9
- overview, 5–6
- running the application, 9–12
- testing the application, 12–13

filters

- action, 115
- ASP.NET MVC and. *See ASP.NET MVC, filters*
- authorization, 317–318
- custom action, 316–317
- exception, 319
- naming, 321–322
- ordering, 320–321
- overview, 305
- summary, 323

focus events, vs. blur events, 131

foreign key relationships, between tables, 18–20

form(s)

- authentication, 111–114
- post values, 64–66
- submitting with jQuery, 290–291

Forms Authentication, 110

Fowler, Martin, 376

frameworks

- for Dependency Injection, 158–159, 389
- mocking, 160–161, 183

frameworks, MVC

- ASP.NET MVC as, 188
- Django, 169
- Ruby on Rails, 168
- Spring, Struts, and Java, 169–170
- Zend Framework and PHP, 170

future-proofing, 371–373

G

global variables, and static methods, 375–376

GUIs (graphical user interfaces)

- history of, 225–226
- JSF and, 169

Guthrie, Scott, 170–171, 278

H

HandleErrorAttribute, 310–311

HandleErrorInfo, 311

Hansson, David Heinemeier, 168

helper methods

- AddRuleViolations, 70–71
- HTML. *See HTML helper methods*
- HomeController, **ClubSite MV, 413**
- HTML helper methods**
 - Html.ActionLink and Html.RouteLink, 258–259
 - Html.BeginForm, 259–260
 - Html.DropDownList and Html.ListBox, 260–261
 - Html.Encode, 260
 - HtmlHelper class and, 257
 - Html.Hidden, 260
 - Html.Password, 261
 - Html.RadioButton, 261–262
 - Html.RenderPartial, 262
 - Html.TextArea, 262–263
 - Html.TextBox, 263–264
 - Html.ValidationMessage, 264–265
 - Html.ValidationSummary, 265–266
 - integration with ModelState collection, 68–69
 - overview, 257
 - patterns of, 258
- Html.ActionLink **helper method, 258–259**
- Html.AttributeEncode, **338–339**
- Html.BeginForm **helper method**
 - editing Dinners with, 61–62
 - implementing, 259–260
- Html.DropDownList **helper method, 260–261**
- Html.Encode **helper method, 260**
- HtmlHelper **class, 257**
- Html.Hidden **helper method, 260**
- Html.ListBox **helper method, 260–261**
- Html.Password **helper method, 261**
- Html.RadioButton **helper method, 261–262**
- Html.RenderPartial **helper method**
 - implementing, 262
 - partial view templates and, 96–97
- Html.RouteLink **helper method**
 - adding page navigation, 108
 - implementing, 258–259
- Html.TextArea **helper method, 262–263**
- Html.TextBox **helper method**
 - editing Dinners with, 62
 - implementing, 263–264
- Html.ValidationMessage **helper method**
 - editing Dinners with, 69
 - implementing, 264–265
- Html.ValidationSummary **helper method**
 - editing Dinners with, 70
 - implementing, 265–266
- HTTP verbs, **differentiating, 64**
- HttpContextBase, **359**

HTTP-GET action method
 creating Dinners with, 72–74
 deleting Dinners with, 78–80
 editing Dinners with, 57–61

HttpOnly, 344

HTTP-POST action method
 creating Dinners with, 74–78, 116
 deleting Dinners with, 80–82
 editing Dinners with, 62–66, 116–118
 for sharing data, 402–403

HttpRequest validation, 342–343

HttpRequestBase
 testing routes and, 359–360
 uploading files and, 419–420

I

IController
 Dependency Injection and, 390
 implementing, 227–228

IDinnerRepository
 extracting, 152–154
 updating DinnersController and, 154

images
 adding with ClubSite, 410–411
 working with ClubSite, 418–420

Index action methods
 adding to DinnersController class, 36–37
 recap, 102

Index view template
 HTML listing using, 41–42
 implementing, 50

inserting
 Dinner rows, 16–17
 with the DinnerRepository class, 29–30

integration tests, 382–383

interface(s)
 extracting IDinnerRepository, 152–154
 filter, 311–312
 future-proofing with, 371–373
 graphical user, 169, 225–226
 implementing IController, 227–228
 IView, 269
 IViewEngine, 268

Inversion of Control (IoC) containers
 for Dependency Injection, 158–159
 StructureMap as, 389–392

IQueryable<T>
 implications of, 110
 understanding, 102–103

Iron Ruby, 168

IsUserRegistered helper method
 enabling RSVPs, 120–121
 Register action method and, 122–123
 running tests and, 150

J**Java frameworks, 169–170****JavaScript**

AJAX and, 278–279
 creating a map Partial view and, 127–129
 Map.js utility library and, 129
 Object Notation. *See JSON (JavaScript Object Notation)*
 RSVP support and, 125–126

JavaScriptResult, 235

description of, 237

jqModal code, 297–299**jQuery**

adding animation to RSVP support, 125–127
 calling the JSON-based AJAX method with, 141–145
 filtering data with a Select box, 295–296
 integrating Map support and, 132
 jqModal code, 297–299
 Microsoft including, 277
 modal popup with, 296–297
 rating control, 299–302
 updating an HTML element with, 290–291

JSF, 169–170**JSON (JavaScript Object Notation)**

calling, 141–145
 implementing, 140–141

JsonResult, 235

description of, 236–237

K**Kothari, Nikhil, 289–290****L****LAMP (Linux/Apache/MySQL/PHP) platforms, 168****libraries**

JavaScript. *See jQuery*
 Map.js utility, 129–130
 referencing required, for MVC, 394–395
 script library references, 123–124

LINQ to SQL

adding classes, 21–22
 creating data model classes with, 22–25
 IQueryable<T> and, 102–103
 overview, 21

M**maintainability approaches, 185–186****Map.js file**

creating a map Partial view, 127–128
 creating a utility library, 129–130

mapping, AJAX support

calling a JSON-based AJAX method, 141–145
 creating a Map.js utility library, 129–130

mapping (continued)

mapping, AJAX support (continued)

- creating a Partial view, 127–129
- implementing a JSON-based AJAX method, 140–141
- implementing location search, 136–140
- integrating with Create and Edit forms, 131–134
- integrating with Details view, 134–136
- overview, 127

master page templates

- functions of, 98–101
- introduction to, 44
- overview, 92–93

matching, route, 217

MCV. See Model-View-Controller (MVC)

MembershipController, ClubSite MVC, 413

method calls, serving in MVC, 171

Microsoft ASP.NET AJAX

- how it works, 288–290
- implementing auto-complete, 292–295
- overview, 288
- Script #, 289–290

migrating Web Forms to MVC. See Web Forms to MVC, migrating

Mitnick, Kevin, 330–331

mocking frameworks

- for testing, 183
- User.Identity.Name property and, 160–161

modal popup windows, 296–297

Model Binders

- overview, 244–245
- using UpdateModel, 245–246
- using validation with, 246–247
- whitelist form binding and, 347–348

model binding security, 82–83

model function

- in ASP.NET MVC, 167, 190–191
- in MVC, 166

model-related tests, 147–149

models, building

- adding LINQ to SQL classes, 21–22
- creating a DinnerRepository class, 26–28
- creating data model classes, 22–25
- integrating validation/business rule logic, 30–34
- LINQ to SQL and, 21
- NerdDinnerDataContext class, 25–26
- overview, 20–21
- using the DinnerRepository class, 28–30

models, creating, 377–379

ModelState collection

- Html.ValidationSummary and, 70
- integration with HTML helper methods, 68–69

Model-View-Controller (MVC)

- definition, 165–166
- Norwegian shipping and, 166–167

Model-View-Controller and ASP.NET

- ASP.NET MVC: the new kid. See ASP.NET MVC, the new kid

- MVC on the Web. See Model-View-Controller on the Web
- overview, 165
- summary, 174
- what Model-View-Controller is, 165–167

Model-View-Controller on the Web

- Django and Python, 169
- MonoRail, 170
- overview, 167–168
- Ruby on Rails, 168
- Spring, Struts, and Java, 169–170
- Zend Framework and PHP, 170

MonoRail, and NVelocity, 272–273

Moq, 160–161

MVC applications, adding Web Forms

- avoiding routing collisions, 399–401
- doing nothing, 399–401
- overview, 398–399
- using System.Web.Routing, 401–402

N

named routes, 206, 211

naming filters, 321–322

navigation, page, 106–109

NerdDinner

- AJAX enabling RSVPs. See AJAX, enabling RSVPs
- authentication and authorization.
 - See authentication; authorization
- building the model. See models, building
- controllers and views. See controllers, NerdDinner application; views, NerdDinner application
- create, update, and delete (CRUD) form scenarios for.
 - See Dinners, CRUD support for
- creating the database. See Database, NerdDinner application
- directory structure, examining of, 6–9
- integrating an AJAX map. See AJAX map, integrating
- overview, 1–5
- pagination support. See pagination support
- partials and master pages. See master page templates; partial view templates
- selecting FilePNew Project. See FilePNew Project, NerdDinner application
- summary, 164
- unit testing. See unit testing, NerdDinner
- ViewData and ViewModel.
 - See ViewData; ViewModel

NerdDinner directory structure, 6–9

NerdDinnerDataContext classes, 25–26

NerdDinner.Tests project, 146–147

NewsController, ClubSite MVC, 413

NHaml, View engine, 270–272

[NonAction] public methods, 346–347

NotFound view template, 42–44

NVelocity View engine, 272–273

O

orchestration, vs. composing, 184
ordering, filter, 320–321
 @OutputCache **directive, 309**
 OutputCacheAttribute
 overview, 308
 properties of, 308–309
 usage examples, 309–310
overflow parameters, in URL generation, 214–215

P

page navigation UI, 106–109
 page **value, 103–106**
paging support
 adding page navigation UI, 106–109
 adding "page" value to the URL, 103–106
 Index action method recap, 102
 overview, 101
 understanding IQueryable<T>, 102–103
parameters
 catch-all, 206–207
 Controller class actions and, 231–232
 Controller class actions and multiple, 233
 mapping with ControllerActionInvoker, 243
 overflow, in URL generation, 214–215
Partial view templates
 AJAX and, 284–287
 clarifying code and, 97–98
 creating a map, 127–129
 creating a RSVP, 127
 Dry Principle and, 93–97
 Html.RenderPartial and, 262
 overview, 92–93
PartialViewResult, 235
 description of, 238
passive injection, 334–336
patterns, URL, 199–202, 204
persistent cookies, 343
Peters, Andrew, 270
 PhotosController, **ClubSite MV, 413**
PHP, 170
posted values, retrieving, 64–66
Poulson, Kevin, 328–329
PRG (Post-Redirect-Get) pattern, 246
Principles of Enterprise Application Architecture (Fowler), 376
public methods, [NonAction], 346–347
Python, 169

Q

querying, DinnerRepository class, 28–29
querystring value, 104

R

Rails, 168
rating control, AJAX, 299–302
reactions, to ASP.NET MVC, 187–188
RedirectResult, 235
 description of, 237
RedirectToRouteResult, 235, 237
refactoring, code, 352
regions, content, 43
 Register **action method**
 calling, 123–125
 implementing, 122–123
repository, NerdDinner data, 136–140
Repository patterns
 creating, 26–28
 creating the model, 377–379
 implementing a stub, 380–381
 implementing the real thing, 382–383
 Northwind Product Repository, 379–380
 overview, 376–377
request lifecycle, 196
result filters
 custom, 314–315
 function of, 311–312
 OutputCacheAttribute as, 308–310
 ResultExecutedContext, **315**
 ResultExecutingContext, **314–315**
 RoutablePage **abstract base class, 224**
 RouteData, **217**
routes, testing, 358–360
routes and URLs
 advanced routing, 217–218
 introduction to routing. *See routing, introduction*
 overview, 197–198
 route extensibility, 218–221
 summary, 224
 tying URLs to actions, 216–217
 using routing with Web Forms, 222–224
routes generating URLs
 ambient values, 211–213
 ambient values and default values, 213–214
 detailed look at, 209–211
 high level view of, 209
 named routes, 211
 overflow parameters, 214–215
 overview, 209
 with the Route class, 215–216
routing
 avoiding collisions, 399–401
 for ClubSite migration to MVC, 411–413
 understanding, in MVC, 37–39
routing, introduction
 catch-all parameter, 206–207
 compared to URL rewriting, 199, 209
 constraints, 205–206

routing (continued)

routing, introduction (continued)

- defining default values, 202–205
- defining route URLs, 199–202
- generating URLs. *See routes generating URLs*
- named routes, 206
- overview, 198–199
- `StopRoutingHandler`, 208

RSVP support

- adding a jQuery animation, 125–127
- calling `Register` action method, 123–125
- creating a RSVP partial view, 127
- implementing `Register` action method, 122–123
- indicating if RSVP'd, 120–122

`RSVPController` class, 122–123

Ruby, 168

S

sanitizing user input, 248–249

scaffolding

- Django and, 169
- of a view, 45, 50

Script #, 289–290

scripting

- cross-site. *See cross-site scripting (XSS)*
- disabled, handling with AJAX, 280–284
- JavaScript. *See JavaScript*

security, application

- cookie stealing threat, 343–344
- cross-site request forgery threat. *See cross-site request forgery (CSRF)*
- cross-site scripting threat. *See cross-site scripting (XSS)*
- error reporting and the stack trace, 345
- knowing the enemy's mind. *See enemy's mind, knowing*
- overview, 325–326
- preventing cookie theft, 344
- securing Controllers, not routes, 345–346
- summary, 348
- using `[NonAction]` to protect public methods, 346–347
- weapons. *See weapons*
- whitelist form binding, 347–348

security, model binding, 82–83

Select box, 295–296

separation of concerns

- meaning of, 184–185
- purpose of, 40, 86

server control replacement, 415–418

server load, 278

server-side controls

- lack of, 187–188
- in Web Forms, 173

Service Layer, 383–385

Session, ASP.NET, 403–404

session cookies, 343

Single Responsibility Principle (SRP), 373

singletons

- performance myth, 375
- tight coupling and, 373–375

social engineering, 330–331

Spolsky, Joel, 368

Spring Framework, 169–170

SQL Server Express database, 14–15

Srizbi Trojan, 333

SRP. *See Single Responsibility Principle (SRP)*

stack trace, 345

StackOverflow.com, 343–344

state management, in Web Forms, 173

static methods, and global variables, 375–376

Storm botnet, 332–333

string concatenation, 364

strongly typed Views, 255–256

`StructureMap`, 389–392

stubs, 380–381

styling, migrating to MVC, 410–411

`System.Web.Routing`, 401–402

T

tables

- adding data to, 20
- creating in SQL Server Express database, 15–18
- foreign key relationships between, 18–20

TDD. *See Test Driven Development (TDD)*

TDD, application, 362–364

- overview, 357
- redirecting to another action, 360–361
- summary, 365
- testing Controllers, 360
- testing routes, 358–360
- testing view helpers, 362–364
- testing views, 364–365

TDD, introduction

- benefits of writing tests, 357
- default unit tests, 353–355
- getting started, 357
- not crossing boundaries, 353
- only test the code you write, 355–357
- overview, 350
- refactoring the code, 352
- repeating, 352–353
- writing a failing unit test, 350–351
- writing a passing unit test, 351–352

`TempData`, 406–407

`TempDataDictionary`, 361

Test Driven Development: *By Example* (Beck), 357

Test Driven Development in Microsoft.NET (Newkirk and Vorntsov), 357

Test Driven Development (TDD)

applying to ASP.NET MVC. *See* *TDD, application introduction*. *See* *TDD, introduction*
overview, 349

testability. *See* *unit testing*

testing routes, 358–360

tests, unit. *See* *unit testing, NerdDinner*

theft, cookie. *See* *cookie theft*

token verification, 342

type basis, for locking down binding, 83

U

Uniform Resource Locators. *See* *URLs (Uniform Resource Locators)*

unit testing

benefits of writing, 357
caring about, 186–187
default tests, 353–355
TDD and. *See* *Test Driven Development (TDD)*
with the TempDataDictionary, 361
Web Forms and, 183
writing a failing test, 350–351
writing a passing test, 351–352
writing good tests, 353–357

unit testing, NerdDinner

creating DinnersController tests, 150–151
creating Edit action tests, 159
creating tests for the Dinner model class, 147–149
creating the FakeDinnerRepository class, 154–157
Dependency Injection and, 152
extracting an IDinnerRepository interface, 152–154
mocking the User.Identity.Name property, 160–161
NerdDinner.Tests project, 146–147
overview, 145–146
running tests, 149–150
summary, 163
testing UpdateModel() scenarios, 162–163
updating DinnersController, 154
using the FakeDinnerRepository class, 157–159

UpdateModel method

Model Binders and, 245–246
model binding security and, 82–83
testing, 162–163

updating, with the DinnerRepository class, 29–30

URL generation

ambient values, 211–213
ambient values and default values, 213–214
detailed look at, 209–211

high level view of, 209

named routes, 211

overflow parameters, 214–215

overview, 209

with the Route class, 215–216

Url.Encode, **338–339**

URLs (Uniform Resource Locators)

adding "page" value to, 103–106
/Dinners/Create , 114–115
embedded values, 104–106
existing authorization, 306
guidelines for, 197
handled by DinnersController, 56–57
querystring values, 104
rewriting of, 199, 209
routes and. *See* *routes and URLs*

usage basis, for locking down binding, 83

user input, sanitizing, 248–249

User.Identity.Name **property**

creating Dinners with, 116
editing Dinners with, 116–118
mocking, 160–161

V**validation**

business rule logic and, 31–34
schema, 30

values

adding page, 103
ambient, 211–214
default, 202–205, 213–214
using embedded, 104–106
using querystring, 104

verbs, HTTP, 64

verification, token, 342

View engines

alternative View engines. *See* *View engines, alternative*
configuring, 267
finding a View, 267–268
overview, 266–267
selecting, 267–268
the View itself, 269

View engines, alternative

Brail, 273–274
NHaml, 270–272
NVelocity, 272–273
overview, 269–270
Spark, 274–275

view function

in ASP.NET MVC, 167, 190–194
in MVC, 166

view helper methods, 362–364

view templates

view templates

- Create, 92–93, 131–134
- Details, 44–49, 134–136
- Edit, 92–94, 131–134
- Index, implementing, 49–54
- NotFound, implementing, 42–44
- partial. *See partial view templates*
- passing data from controllers to, 86–87

ViewContext, **269**

ViewData

- overview, 86–87
- using the dictionary, 87–88

ViewModel

- custom-shaped classes, 92
- overview, 86–87
- pattern, 89–91

ViewResult, 235

- description of, 237

Views

HTML helper methods. *See HTML helper methods*

new View engine or new ActionResult, 275

overview, 251

specifying, 253–255

strongly typed, 255–256

summary, 275

View engine. *See View Engines*

what they do, 251–253

what they shouldn't do, 253

views, NerdDinner application

convention-based naming and the `\Views` directory structure, 54–56

implementing the “Details” view template, 44–49

implementing the “Index” view template, 49–54

implementing the “NotFound” view template, 42–44

with our controller, 40–42

\Views directory structure

in ASP.NET MVC, 54–56

creating a RSVP partial view, 127

ViewState, 183

Virtual Earth

in creating a map Partial view, 128–129

integrating the map, 132

interacting with `Map.js` file, 129

vulnerability, XSS, 344

W

weapons

digital stealth ninja network, 333–334

overview, 331

spam, 331–332

Srizbi Trojan, 333

Storm botnet, 332–333

Web, and MVC. *See Model-View-Controller on the Web*

Web Forms and MVC, sharing data

overview, 402

using ASP.NET Session, 403–404

using Cross-Page Posting, 404–405

using HTTP POST, 402

using TempData, 406–407

Web Forms and MVC together

adding Web Forms to existing ASP.NET MVC

applications. *See MVC applications, adding Web Forms*

how it is possible, 393–394

including MVC in existing Web Forms applications.

See Web Forms applications, including MVC

migrating from Web Forms to MVC. *See Web Forms to MVC, migrating*

overview, 393

sharing data. *See Web Forms and MVC, sharing data*

summary, 420

Web Forms applications, including MVC

creating directories, 395–396

overview, 394

referencing required libraries, 394–395

updating the `Web.config`, 396–398

Web Forms to MVC, migrating

adding images and styling, 410–411

ClubSite MVC Controllers, 413–415

creating an empty ASP.NET MVC project, 407

implementing the structure, 408–410

overview, 407

replacing complex server controls, 415–418

setting up routing, 411–413

uploading files and working with images, 418–420

Web.config, 396–398

whitelist form binding, 347–348

X

XSS. *See cross-site scripting (XSS)*

Z

Zend Framework, 170