

A

- Abstract class 65
- abstract ComposablePartCatalog class 504
- abstract factories 113, 139–140
- Abstract Factory, misconceptions about DI 7
- ABSTRACTIONS 106, 380
 - closing connections through 170–173
 - configuring multiple implementations of same 400–401
 - mapping runtime values to 163–170
 - CurrencyProvider example 168–170
 - design considerations 166
 - selecting routing algorithm example 166–168
 - mapping to concrete types 316–317, 421–422, 497–498
- AbstractLifestyleManager class 328–329, 332
- [AcceptVerbs] attribute 298
- AccountController class 147
- Activator class 150
- Activator.CreateInstance 151–153
- Adapter design 12
- adapters, exporting 501–503
- AddAllTypesOf method 355
- AddExtension method 459
- addIns field 112
- AddNewExtension method 459, 485
- AddRegistry method 360
- AddService 108
- ADO.NET Data Services 38
- AggregateCatalog 507–509
- AllowDefault property 523
- AllTypes class 320
- AMBIENT CONTEXT pattern 118–132
 - caching currency 123–130
 - CachingCurrency class 126–128
 - modifying time 128–130
 - TimeProvider 125–126
 - challenges
 - of implementation 122–123
 - with ASP.NET framework 123
 - description of 118–120
 - example of 123
 - implicitness 122
 - related patterns 130–132
 - when to use 120–121
- anti-patterns 133–161
 - BASTARD INJECTION 144–149
 - analysis of 146–149
 - example of 144–146
 - CONSTRAINED CONSTRUCTION 149–154
 - analysis of 151–154
 - example of 149–151
 - CONTROL FREAK 136–144
 - analysis of 143–144
 - examples of 136–142
 - SERVICE LOCATOR 154–161
 - analysis of 157–161
 - example of 156–157
- AnyConcreteTypeNotAlreadyRegisteredSource 420–421
- AOP (ASPECT-ORIENTED PROGRAMMING) 277, 296
- APIs (Application Programming Interfaces)
 - configuring difficult
 - parts with non-public constructors 522–523
 - primitive DEPENDENCIES 341–343, 412–413, 486–487
 - primitive parts 521–522
 - registering components with code
 - blocks 343–344, 381–382, 487–489
 - static factories 413–414

- APIs (*continued*)
 - wiring with PROPERTY INJECTION pattern 344–346, 382–414, 489–491, 523–525
 - lifestyle 327–328, 365
 - LifetimeManager 464–465
 - registering difficult 442–447
 - configuring primitive DEPENDENCIES 443
 - objects with code blocks 444–445
 - wiring with PROPERTY INJECTION design pattern 445–447
 - App.xaml file 77
 - AppDomain 303
 - application configuration files, loading XML using 394–395
 - Application Programming Interfaces. *See* APIs
 - Application_Start event 77
 - Application_Start method 79–80, 85
 - ArgumentNullException 101, 108
 - ArrayEnumerable<T> class 403, 405, 412
 - arrays, configuring 337–338
 - ASP.NET applications, OBJECT COMPOSITION for 224–230
 - ASP.NET framework, challenges of AMBIENT CONTEXT pattern with 123
 - ASP.NET MVC applications, OBJECT COMPOSITION in 206–210
 - CommerceControllerFactory example 208–210
 - with custom Controller Factory 206–208
 - ASP.NET MVC Controllers 73
 - aspect attribute 296
 - ASPECT-ORIENTED PROGRAMMING. *See* AOP
 - aspects, declaring using attributes 296–300
 - disadvantages of 298–300
 - modifying compilation 296–297
 - using custom host 297–298
 - assembly catalogs 506
 - Assembly instance 355
 - AssemblyCatalog class 506
 - AssemblyContainingType method 355
 - AssemblyNames 193
 - AssemblyResources 393, 395
 - auditing aspect 285
 - AuditingProductRepository class 278–281, 284
 - autodetect value 392
 - autofac configuration 427
 - Autofac DI Container 312, 417–447
 - introduction to 418–429
 - configuring ContainerBuilder 422–427
 - resolving objects 420–422
 - managing DEPENDENCIES with 245–247
 - managing lifetime 429–433
 - multiple components 433–442
 - selecting among multiple candidates 434–438
 - wiring 438–442
 - registering difficult APIs 442–447
 - configuring primitive DEPENDENCIES 443
 - objects with code blocks 444–445
 - wiring with PROPERTY INJECTION design pattern 445–447
 - Autofac.Configuration assembly 426
 - Autofac.IContainer 422
 - AUTO-REGISTRATION 67–68, 73, 355–357, 424–426
 - autowire attribute 392
 - AUTO-WIRING
 - containers 64–67
 - DEPENDENCIES 392–393
 - sequences 374–375, 402–404, 438, 476–477, 516–517
 - with PROPERTY INJECTION pattern 415–416
 - Azure Table Data Access library 39
 - Azure Table Storage Service 38
 - AzureProductRepository 141–142
-
- B**
- Base Class Library. *See* BCL
 - Basket class, converting 114–117
 - basket feature, for complex example 54–57
 - Basket instance 115
 - BasketContainer class 208–209, 233–234
 - BasketController class 62, 65, 108–110
 - BasketController example 65–67
 - BasketController’s Index method 108, 110
 - BasketDiscountPolicy 55–57, 234, 262
 - BasketManager class 233–234
 - BasketPowerShell library 233
 - BasketPowerShellLogic 232
 - BasketRepository class 55, 66, 234
 - BasketService class 55, 57, 70, 75
 - BASTARD INJECTION anti-pattern 144–149
 - analysis of 146–149
 - impact 147–148
 - refactoring toward DI 148–149
 - example of 144–146
 - BCL (Base Class Library) 22, 37, 100, 107, 113
 - BCL modules 194
 - BeginLifetimeScope method 431–432
 - benefits, of DI 15–21
 - extensibility 17–18
 - late binding 15–17
 - maintainability 19–20
 - parallel development 19
 - TESTABILITY 19–20
 - unit testing 21
 - boundary behavior option 267
 - Breeding class 406, 440, 479, 519–520
 - Breeding constructor 441–442
 - Breeding instance 377, 379, 406, 441
 - Breeding object 406

BuilderStrategies 471–472
 BuilderStrategy class 469–471
 BVT (Build Verification Test) 194

C

CacheLifecycle class 367, 369
 CacheLifestyleManager class 329–332
 CacheLifetimeManager class 466–468, 470–472
 CacheLifetimeStrategyExtension 470, 472
 caching
 currency 123–130
 CachingCurrency class 126–128
 modifying time 128–130
 TimeProvider 125–126
 custom lifestyles 328–331, 366–369
 caching aspect 285
 caching lifestyles, developing 465–467
 CachingCurrency class 126–128
 CachingCurrencyProvider 126
 CaesarSalad class 382–383, 445–446, 489–490
 CampaignContainer 226
 CampaignDataSource class 227–229
 CampaignPresenter class 227–229
 CampaignPresenter example 225–230
 CampaignRepository class 229
 Castle Windsor container 313–346
 configuring difficult APIs 341–346
 primitive DEPENDENCIES 341–343
 registering components with code blocks 343–344
 wiring with PROPERTY INJECTION pattern 344–346
 introduction to 314–323
 configuring container 317–322
 packaging configuration 322–323
 resolving objects 315–317
 managing lifetime 323–333
 multiple components 333–341
 selecting among multiple candidates 333–336
 wiring 336–341
 Castle.MicroKernel.Registration.Component 316, 318
 catalogs 504–509
 aggregate 507
 assembly 506
 directory 506–507
 filtering 507–509
 type 505–506
 with containers 504–505
 Certified Microsoft Partner 31
 cheap hotel example, purpose of DI 8–13
 CheckLease method 369
 chicken component 490
 chicken instance 383
 ChiliConCarne class 382, 413, 443–444, 487–488
 CI (Continuous Integration) 194
 Circuit Breaker Interceptor 305–306, 409–410, 483–484
 Circuit Breaker pattern 286–291, 303
 [CircuitBreaker] attribute 296, 299, 307
 CircuitBreaker class 290, 486
 CircuitBreakerInteceptionBehavior 484–485
 CircuitBreakerInterceptor class 306, 308, 409–411
 CircuitBreakerProductManagementAgent class 288, 291, 300
 Close method 171, 249
 cmdlet 230
 coarse-grained ABSTRACTIONS 184
 code
 as configuration 352–354, 423–424, 454–455
 configuring containers with 70–72
 configuring instance scope with 430
 configuring lifestyles with 324, 362–363
 managing lifetimes with 460–461
 code blocks
 configuring objects with 381–382
 registering components with 343–344, 487–489
 registering objects with 444–445
 Code Query Language. *See* CQL
 CodeConfig 397
 CollectionResolver 339
 CommerceControllerFactory example 208–210
 CommerceDomain module 145
 CommerceInstanceProvider 216–217
 CommerceIntegrationTest project 192
 CommerceObjectContext class 33, 36, 40, 43, 49
 CommerceService 279
 CommerceServiceContainer class 216, 243
 CommerceServiceHost class 215
 CommerceServiceHostFactory class 215, 218
 CommerceSqlDataAccess module 145
 CommerceWindsorInstaller class 79, 85–86
 Community Technical Preview. *See* CTP
 Component class 320
 ComponentNotRegisteredException 420
 components
 concrete 480
 configuring with custom lifestyles 332–333, 369–370
 multiple
 Interceptors 407–412, 481–486
 selecting among multiple candidates 333–336, 371–374, 400–402
 wiring 336–339, 402–405, 438–442, 476–480, 516–521
 named 479–480
 picking from larger set 438–440, 477–479
 registering multiple implementations of same 473–475

- components (*continued*)
 - registering with code blocks 343–344, 487–489
 - releasing 325, 431–433, 461–464
 - with custom lifetimes
 - registering 467
 - releasing 467–469
- components element 427
- ComposablePartCatalog class 504–505, 508
- ComposablePartDefinitions 508
- COMPOSER internal 230
- composing
 - difficult APIs 521–525
 - parts with non-public constructors 522–523
 - primitive parts 521–522
 - wiring with PROPERTY INJECTION design
 - pattern 523–525
 - Window example 178–181
- Composite pattern 11, 106
- CompositeNotificationService 188
- COMPOSITION ROOT 53, 78, 81, 200, 203–204
- COMPOSITION ROOT pattern, for containers 75–81
- CompositionContainer 498, 504–505
- conceptual problem 41
- concrete components 480
- concrete contracts, Decorator design pattern
 - with 519–521
- concrete types, mapping ABSTRACTIONS to 316–317, 421–422, 497–498
- ConcreteA 175
- ConcreteC 175
- .config file 153
- ConfigSectionResource 393
- configuration
 - code as 318–319, 352–354, 423–424, 454–455
 - files, loading XML using 394–395
 - packaging 322–323, 358–361, 427–429, 458–459
- ConfigurationExpression class 72, 351, 353, 355, 359, 363
- ConfigurationInstaller 321
- configurationLocations parameter 396
- ConfigurationManager API 456
- ConfigurationSettingsReader 426
- Configure method 353–355, 358–360
- configure parameter 353
- configured instances, picking from larger set 375–377
- configuring, containers 67–75
 - by convention 72–75
 - with code 70–72
 - with XML 68–70
- connectionString 246
- ConnectionStringConverter 70
- connectionStrings element 151
- connString parameter 66, 74
- console applications, OBJECT COMPOSITION
 - for 202–205
- ConsoleMessageWriter class 14–16, 18, 22
- ConsoleMessageWriter instance 27
- CONSTRAINED CONSTRUCTION anti-pattern 135, 149–154
 - analysis of 151–154
 - impact 151–152
 - refactoring toward DI 152–154
 - example of, late-binding ProductRepository
 - class 149–151
- constructor arg elements 392, 406, 413
- CONSTRUCTOR INJECTION 64, 77, 81, 96–97, 206, 216
- CONSTRUCTOR INJECTION pattern 98–103
 - description of 98–99
 - examples of 100–101
 - related patterns 103
 - when to use 99–100
- Constructor Over-injection, refactorings for 182–188
 - OrderService example 185–188
 - refactoring to Facade Services 183–184
- constructor-arg elements 392, 402, 414
- constructors, non-public 522–523
- Consumer class 155
- Container class 349, 359
- Container Controlled 460
- container instance 85
- Container methods 81
- container.GetInstance<IIngredient>()
 - method 371
- container.Resolve<IIngredient>() method 333, 434
- container.ResolveAll<ICourse>() method 477
- ContainerBuilder API 428
- ContainerBuilder class 419–420, 424, 426, 429–430
- ContainerBuilder instance 421, 428
- ContainerBuilder, configuring 422–427
 - AUTO-REGISTRATION 424–426
 - CODE AS CONFIGURATION 423–424
 - packaging configuration 427–429
 - XML configuration 426–427
- ContainerControlledLifetimeManager 460–461
- containers 58–92
 - and Microsoft 89–92
 - AUTO-WIRING 64–67
 - catalogs with 504–505
 - configuring
 - .NET types in XML 390–391
 - AUTO-REGISTRATION 319–320, 355–357, 455–456
 - by convention 72–75
 - CODE AS CONFIGURATION 318–319, 352–354, 454–455

- containers (*continued*)
 - DEPENDENCIES 391–393
 - with code 70–72
 - XML 68–70, 321–322, 357–358, 456–458
 - disposing 512–513
 - history of 89–92
 - managing DEPENDENCIES with 242–247
 - using Autofac 245–247
 - using specialized container 243–244
 - misconceptions about DI 7–8
 - patterns for 75–87
 - COMPOSITION ROOT pattern 75–81
 - REGISTER RESOLVE RELEASE pattern 81–87
 - resolving controllers with various 62–63
 - resolving object graphs with 63–64
 - selecting 87–89
 - Contains method 376
 - ContentWindow 179
 - context field 389
 - context parameter 113
 - Continuous Integration. *See* CI
 - ContractMapper 243, 245–246
 - contracts, Decorator design pattern with 519–521
 - CONTROL FREAK anti-pattern 136–144
 - analysis of 143–144
 - examples of
 - factories 137–142
 - newing up DEPENDENCIES 136–137
 - Controller Factory, custom 206–208
 - creating 207–208
 - registering 208
 - controller instance 87
 - ControllerBuilder 208
 - controllerFactory instance 208
 - controllerName 207
 - controllerType 209, 352, 389, 422, 453, 498
 - Convention method 356
 - conventions
 - configuring containers by 72–75
 - resolving named components by 437–438
 - ConvertTo method 115–117
 - CordonBleu class 403, 405, 438, 515, 519
 - correct type attribute 406
 - Cotoletta 440
 - coupling, monitoring 188–196
 - integration-testing coupling 191–193
 - unit-testing coupling 189–191
 - using NDepend 193
 - Course object 415
 - courses constructor 337
 - courses, refactoring sequences to better 337
 - CQL (Code Query Language) 194
 - Create method 181, 343, 381
 - CreateChild method 178, 180
 - CreateChildContainer method 463
 - CreateController 51, 207
 - CreateDefaultRepository method 146
 - CreateInstance method 150
 - CreateServiceHost method 218
 - creation policies
 - declaring 509–511
 - exporting 510
 - importing 510–511
 - importing with requirements of 510–511
 - CreationPolicy enum 509
 - CreationPolicy.Any 510
 - CROSS-CUTTING CONCERNS 118, 276, 285–295, 308
 - adding security 293–295
 - Circuit Breaker design pattern 286–291
 - handling exceptions 292–293
 - Ctor method 72, 373, 378
 - Ctor/Is method 378
 - Ctor<spiciness>() method 381
 - Ctor<T> method 378–381, 383
 - CTP (Community Technical Preview) 91
 - currency
 - caching 123–130
 - CachingCurrency class 126–128
 - modifying time 128–130
 - TimeProvider 125–126
 - implementing 116–117
 - injecting 115–116
 - providers, adding to shopping basket 101–103
 - Currency class 101, 123, 202, 204
 - Currency instance 114, 168, 205
 - Currency type 168
 - CurrencyContainer class 203–204
 - CurrencyParser class 203–205
 - CurrencyProfileService 102, 108–109, 169
 - CurrencyProvider class 101–103, 126, 139, 204, 208
 - CurrencyProvider example 168–170
 - CurrencyUpdateCommand 204–205
 - current field 125
 - Current property 119
 - Cutlet object 406
 - outlet variable 377
- ## D
-
- Data Access Layer, complex example 37
 - right way 49–50
 - wrong way 32
 - data binding 221
 - Data Transfer Object. *See* DTO
 - DataContext property 178, 180–181, 220–222
 - DataContract 213
 - DateTime 124, 129
 - DateTime members 120
 - DateTime.Now 128, 332
 - DateTime.UtcNow 124–125

- DDOS (Distributed Denial of Service) 286
- Decorator design pattern
 - concrete component 480
 - named component 479–480
 - wiring
 - explicitly 339–340
 - implicitly 340–341
 - with concrete contracts 519–521
 - with delegates 378–380, 441–442
 - with inline objects 406–407
 - with instance references 377–378
 - with named instances 378
 - with named objects 406
 - with WithParameter method 440–441
- defaultAssembly attribute 427
- default-autowire attribute 392
- DefaultContext 119
- DefaultControllerFactory class 207–209, 389, 422, 498
- DefaultControllerFactory.GetControllerInstance 209
- DefaultCurrencyProfileService 108
- DefaultProductDiscountPolicy 56
- DefaultTimeProvider class 126
- DefaultWorkflowLoaderService 108
- DefaultWorkflowSchedulerService class 108
- delegates, Decorator design pattern with 378–380, 441–442
- DeleteProduct method 250, 301
- DEPENDENCIES 24–26, 37, 136–137, 143, 149, 151, 216, 338, 496
 - configuring
 - AUTO-WIRING 392–393
 - explicitly 391–392
 - named 372–373, 401–402, 475–476
 - cyclic 175–181
 - breaking cycle with PROPERTY INJECTION 177–178
 - composing Window example 178–181
 - lifetime for 239–247
 - disposable 247–255
 - managing with container 242–247
 - primitive 341–343, 380–381, 412–413
 - registering 335–336, 435–437
 - short-lived 170–175
- Dependency Injection. *See* DI
- Dependency Inversion Principle. *See* DIP
- DEPENDENCY LIFETIME MANAGEMENT 238–239
- DependsOn method 342–343, 345
- Design Patterns 96
- DesiredService 273
- DI (Dependency Injection) 3–28
 - benefits of 15–21
 - extensibility 17–18
 - late binding 15–17
 - maintainability 19–20
 - parallel development 19
 - TESTABILITY 19–20
 - unit testing 21
 - Hello World example 13–21
 - misconceptions about 5–8
 - is Abstract Factory 7
 - only for late binding 6
 - only for unit testing 6
 - requires container 7–8
 - purpose of 8–13
 - refactoring toward
 - BASTARD INJECTION anti-pattern 148–149
 - CONSTRAINED CONSTRUCTION anti-pattern 152–154
 - CONTROL FREAK anti-pattern 143–144
 - SERVICE LOCATOR anti-pattern 159–161
 - scope of 24–28
 - INTERCEPTION 26–27
 - OBJECT COMPOSITION 25–26
 - OBJECT LIFETIME 26
 - three dimensions of 27–28
 - SEAMS 22
 - STABLE DEPENDENCIES 23
 - VOLATILE DEPENDENCIES 23–24
- DI CONTAINERS 88–89, 198, 202–203, 239
- DI patterns 225
- different route-calculation algorithms 166
- DIP (Dependency Inversion Principle) 285
- DirectoryCatalog class 506–507
- disadvantages, of declaring aspects using
 - attributes 298–300
- DiscountCampaign 241, 259, 261–262
- DiscountedProduct class 46
- discountPolicy 66
- DiscountRepository class 98, 164, 259, 262–263
- DiscountRepository instance 241
- DispatchRuntime 217
- disposable DEPENDENCIES, lifetime for 247–255
 - consuming 248–251
 - managing 251–255
- Dispose method 171, 249, 432, 465
- disposing containers 512–513
- Distributed Denial of Service. *See* DDOS
- Domain Model Layer, for complex example
 - right way 45–49
 - wrong way 32–34
- Domain Object Product 213
- Domain-Driven Design 122, 214
- DomainModel library 153
- DomainModel module 190
- DoStuff method 112
- DTO (Data Transfer Object) 213–214
- dynamic INTERCEPTION 300–303, 308
- dynamic structure, for containers 84–85

E

EggYolk class 64
 EggYolk elements 391, 393
 EggYolkAdapter 503
 embedded resources, loading XML using 395
 EndRequest event 264
 EnrichWith method 379
 EnsureInitialize method 181
 Entity Data Model Wizard 32
 EntityClient provider 36
 EntityDataAccess library 153
 entrée1 variable 376
 EnumerableOf method 376
 EnumerableOf/Contains method 376
 error handling 285
 ErrorHandlingInterceptionBehavior class 482, 485
 ErrorHandlingInterceptor class 305, 408, 410–411
 ErrorHandlingProductManagementAgent class 292–293, 298–299, 304
 examples, complex 29–57
 extending application 53–57
 architecture 53–54
 basket feature 54–57
 right way 41–53
 analyzing loosely coupled implementation 51–53
 Data Access layer 49–50
 Domain Model Layer 45–49
 User Interface Layer 43–45
 wrong way 30–41
 analyzing issues 39–41
 Data Access Layer 32
 Domain Model Layer 32–34
 evaluating application 37–39
 User Interface layer 34–36
 Except method 425
 exception handling Interceptor 303–305, 407–408, 481–483
 exceptions, CROSS-CUTTING CONCERNS 292–293
 ExpensiveService 272
 [Export] attributes 497–498, 500–503, 519
 [Export] overload 500
 [Export] property 502
 exporting, with creation policies 510
 ExportProviders 505
 exports, adapters 499–503
 multiple implementations of same 514–515
 named, importing 515–516
 picking from larger set 517–519
 releasing 512
 types 500–501
 extensibility, benefits of DI 17–18
 Extensible Markup Language. *See* XML

Extent<EvaluatedProduct> 55
 Externally Controlled 460
 Extra property 382–383, 445

F

Facade Services, refactoring to 183–184
 factories 137–142
 abstract 139–140
 static 140–142, 413–414
 factory-method 414
 fault tolerance 285
 FeaturedProductsViewModel 44, 46, 52
 files, loading XML using 394–395
 FileSystemResource 393–394
 FillAllPropertiesOfType method 383
 filtering catalogs 507–509
 FindCache method 365, 367
 FirstOrDefault 400
 Fluent Registration API 319
 For method 319
 For/Use method 374, 381
 For/Use statement 382
 foreach loops 96, 190
 FOREIGN DEFAULT, ProductService class with 144–146
 FromAppConfig method 321
 FromAssemblyContaining generic method 320
 full-fledged DI CONTAINER 91
 Func<IMeal> 381
 Func<IUnityContainer> 488
 Future objects, lifetime for 272
 FutureService 273

G

Get method 366, 368
 GetAllInstances 371
 GetBasketCmdlet class 232, 234
 GetBasketCmdlet example 231–235
 GetControllerInstance method 80, 86–87, 209, 351, 389
 GetCurrencyCode method 108, 169
 GetExchangeRateFor method 114, 116, 123, 127, 129, 168
 GetExport method 497, 500, 512
 GetExportedValues methods 497–498, 500, 515
 GetExports method 498
 GetFeaturedProducts method 40, 46, 48, 52, 137
 GetInstance methods 211, 217, 351–352, 361, 372
 GetInstance<T> method 350
 GetMessage method 122
 getNext method 482

getNext parameter 482
 GetObject method 388, 397, 400
 GetObjectsOfType method 389, 400–401, 414
 GetRequiredInterfaces method 482
 GetRoute method 167
 GetService method 108, 156
 GetService<T> method 159
 GetValue method 465, 467
 Global.asax 51, 77, 79
 GridView control 226, 228

H

[HandleError] attribute 296, 298
 HasExcessCapacity property 269
 HasInterceptors method 307
 Hello World example 13–21
 HierarchicalLifetimeManager 463, 468–469, 471
 history, of containers 89–92
 HomeController class 46–48, 69–70, 208–209
 HomeController, with request-shared repository
 example 262–265
 HttpContext 33, 123, 207, 361, 364
 HttpContext.Current 228, 263–264
 HttpContext.Current.Items 364
 HttpRequestBase 166
 HttpSession 361
 Hybrid 361
 HybridHttpSession 361

I

IActivatingEventArgs<T> interface 446
 IApplicationContexts 398–399
 IAssemblyScanner instance 355
 IAssemblyScanner interface 363
 IAuditor interface 278–279, 284
 IBasketService class 116, 233–234
 IBasketService interface 56, 66, 70, 72, 75
 IBillingSystem 185
 IBuilderPolicy instances 464
 ICircuitBreaker DEPENDENCY 300, 305–306,
 409–411
 ICircuitBreaker instance 409
 ICircuitBreaker interface 290–291, 301, 483–485
 IClientContractMapper 222
 ICollection interface 400
 ICommand 205
 ICommerceServiceContainer 256, 268
 ICommunicationObject 267
 IComponentActivator 328, 330
 IComponentContext 422, 436–437, 440–441, 446
 IContractBehavior 216–217
 IContractMapper 213, 218

IController instance 317
 IControllerFactory interface 56–57, 206–209
 IControllers 62, 73, 86–87
 ICourse
 contract 500, 518
 DEPENDENCIES 401–402, 438, 454, 489–490
 implementations 402
 instances 358, 369–370, 374, 376, 438
 interface 319, 338, 345, 500
 mappings 335, 372, 435, 475
 objects 403
 registrations 477
 ICourse components 424, 438, 440, 444–446
 IDependencyResolver 207
 IDesigner 113
 IDesignerHost implementations 113
 IDictionary 336
 IDisposable interface 239–240, 247–249, 272
 IDisposable pattern 171–172, 174–175
 IEnumerable<ICourse> 376, 402, 404, 440, 517
 IEnumerable<ICourse> DEPENDENCY 476–477
 IEnumerable<ICourse> instance 337–338
 IEnumerable<INotificationService> 188
 IEnumerable<T> 338–339, 403, 434, 476, 501
 IFoo implementation 165
 IFoo instances 165
 IFooFactory 165, 173
 IHttpModule 264
 Ingredient
 components 434
 constructor parameter 377
 exports 514
 implementations 424, 428, 455
 instance 455, 467, 479–480, 489
 interface 318–319, 500–501, 519–520
 objects 401, 415
 properties 383
 services 425, 441
 type 383, 455
 InstanceExpression<ICourse> 373
 InstanceProvider 211, 213, 215–217, 253
 IInterceptionBehavior interface 481–482
 IInterceptor interface 303–306
 InventoryManagement 185–186
 IInvocation interface 304
 IKernel instance 328
 ILease
 instance 367, 466
 interface 466
 member 368
 ILifecycle interface 365–367
 ILifestyleManager 328–329
 ILifetimePolicy instance 365, 465, 470
 IList<Ingredient> 371
 IList<T> 476

- IListableObjectFactory interface 389
- ILocationService 185–186
- IMainWindowViewModelFactory 223
- IMeal configuration 376
- IMeal services 337, 339, 343
- IMessageService 185
- IMessageWriter interface 14–15, 17–18, 21–22
- IMessageWriter.Write method 21
- IMethodInterceptor interface 407–408
- IMethodInvocation interface 408
- IMethodReturn 484
- IModelBinder interface 114
- IModelInterceptorsSelector interface 307, 411
- IModule interface 428
- ImplementedBy method 316, 319
- ImplementedContracts 216
- implicitness, of AMBIENT CONTEXT pattern 122
- [Import] attribute 504–505, 510, 515
- Import.Any 511
- Import.NonShared 511
- Import.Shared 511
- importing
 - named exports 515–516
 - with creation policy requirements 510–511
- [ImportingConstructor] attributes 503–504, 510, 515–516, 521
- [ImportMany] attribute 517–519
- imports 499–500, 503–504
- Include method 356
- Index method 46, 103
- Index View 36
- ingredient constructor parameter 341
- ingredient variable 512
- IngredientExtension class 458
- IngredientInstaller 322
- IngredientModule 428
- ingredients dictionary 401
- Init method 328–330
- Initialize method 113, 458
- InjectionConstructor class 477–480, 487, 489
- InjectionFactory 488
- InjectionMember 475, 485, 489–490
- InjectionProperty class 489
- inline objects, Decorator design pattern with 406–407
- innerAgent 289
- INotificationService interface 187–188
- InputStreamResource 393, 396
- InsertProduct methods 293–294, 301, 304
- Install method 322
- instance references
 - Decorator design pattern with 377–378
 - wiring 373–374
- instance scope, configuring 430–433
 - releasing components 431–433
 - with code 430
 - with XML 431
- Instance variables 376
- InstanceContextMode 211
- InstancePerDependency methods 433
- InstancePerLifetimeScope method 433
- integration-test project 191
- integration-testing coupling 191–193
- Intention-Revealing Interfaces 122
- Intercept method 304
- INTERCEPTION 275–309
 - AuditingProductRepository example 277–281
 - CROSS-CUTTING CONCERNS 285–295
 - adding security 293–295
 - Circuit Breaker design pattern 286–291
 - handling exceptions 292–293
 - declaring aspects 295–309
 - dynamic 300–303
 - patterns for 281–285
 - scope of DI 26–27
 - Windsor example 303–309
 - activating Interceptors 306–309
 - Circuit Breaker Interceptor 305–306
 - exception handling Interceptor 303–305
- Interceptor<InterfaceInterceptor> 485
- Interceptors 481–486
 - Circuit Breaker 409–410, 483–484
 - configuring 410–412, 484–486
 - exception handling 407–408, 481–483
- interface 9, 65
- Interface Segregation Principle. *See* ISP
- InvalidOperationException 484
- Inversion of Control. *See* IoC
- Invoke method 408
- IObjectCache interface 365–368, 465
- IObjectFactory interface 388–389
- IoC (Inversion of Control) 42
- IOrderFulfillment interface 186
- IPresentationMapper interface 229
- IPrincipal interface 48, 295
- IProduct 53
- IProductChannelFactory 250
- IProductManagementAgent interface 249–250, 288–289, 298–300
- IProductManagementAgent object 410
- IProductManagementService 213, 243, 246, 254, 256, 258
- IProductManagementServiceChannel 250
- IRegistration 318
- IRegistrationConvention interface 356, 363
- IRegistrationSource 420–421
- IResource instance 396
- IResource interface 172, 396
- IRouteAlgorithmFactory 167–168
- Is method 378

isCustomerPreferred parameter 34
 IService DEPENDENCY 272–273
 IService interface 155
 IService.SelectItem() 272
 ISomeInterface 104–105, 111
 ISomeRepository 153
 ISomeService interface 152–153
 ISomeServiceFactory 152–153
 ISP (Interface Segregation Principle) 284
 IsPreferredCustomer property 33
 ITypeDescriptorContext 113
 IUnityContainer class 452–453, 461, 463
 IWindow 178–180, 221, 223
 IWindsorContainer 305
 IWindsorInstaller interface 322

J

JunkFood class 343, 381–382, 413, 444, 488
 JunkFood object 414
 JunkFoodAdapter class 523
 JunkFoodFactory class 343, 381, 413, 444, 522
 JunkFoodFactory.Create method 444, 523

K

KeyNotFoundException 156

L

late binding 16
 benefits of DI 15–17
 misconceptions about DI 6
 ProductRepository class 149–151
 late-bound services 320
 Lazy objects, lifetime for 271
 Lazy<IIngredient> 512
 LazyService 255, 272
 Lease property 330
 LeasedObjectCache class 367
 LeasedObjectCache constructor 368
 leases, implementing 331–332
 LifecycleIs method 362
 lifestyle API 365
 Lifestyle property 324
 lifestyles
 advanced 325–327
 PerWebRequest lifestyle 327
 Pooled lifestyle 325–327
 caching, developing 465–467
 configuring 324–325, 362–364
 releasing components 325
 with code 324, 362–363
 with XML 324–325, 363–364

custom 327–333, 364–370
 caching 328–331
 caching lifestyle 366–369
 configuring components with 332–333,
 369–370
 implementing lease 331–332
 lifestyle API 327–328, 365
 preventing memory leaks 364
 lifetime
 for DEPENDENCIES 239–247
 disposable 247–255
 managing with container 242–247
 for Future objects 272
 for Lazy objects 271
 for PER GRAPH objects 259–261
 sharing repository within graph
 example 260–261
 when to use 260
 for POOLED objects 266–271
 reusing expensive repositories example
 268–271
 when to use 267
 for SINGLETON objects 255–258
 thread-safe in-memory repository
 example 256–258
 when to use 256
 for TRANSIENT objects 258–259
 resolving multiple repositories example
 258–259
 when to use 258
 for WEB REQUEST CONTEXT objects 261–266
 and Session Request Context objects 265
 and Thread Context objects 265–266
 HomeController with request-shared repository example 262–265
 when to use 261–262
 LIFETIME MANAGEMENT 27, 198
 lifetime scope 431
 LifetimeManager API 464–465
 LifetimeManager class 460–461, 463–467
 lifetimes, managing 361–370
 configuring 460–464
 configuring instance scope 430–433
 configuring object scopes 398–399
 declaring creation policy 509–511
 developing custom 464–473
 lifestyles 324–333, 362–364
 releasing objects 511–513
 LifetimeStrategy class 469–471
 LISKOV SUBSTITUTION PRINCIPLE. *See* LSP
 LoadConfiguration method 70, 456
 LobsterBisque class 376, 405, 518–519
 Locator class 156, 158
 Locator.Reset() method 159
 logging aspect 285

loosely coupled model 9
LSP (LISKOV SUBSTITUTION PRINCIPLE) 281, 284

M

Main method 14, 202–203, 205, 251
MainObjectCache 367–368
maintainability 16, 19–20
MainViewModel 249
MainWindow 221, 223–224
mainwindow variable 180–181
MainWindowAdapter 181, 222–224
MainWindowViewModel class 174, 180, 221–222
MainWindowViewModelFactory 222–223
Managed Extensibility Framework. *See* MEF
mapping, ABSTRACTIONS to concrete types 316–317, 421–422, 497–498
maximum size option 267
MaxSize property 269
MayonnaiseAdapter 503, 506
MEF (Managed Extensibility Framework) 88, 91, 492–525
 composing difficult APIs 521–525
 parts with non-public constructors 522–523
 primitive parts 521–522
 wiring with PROPERTY INJECTION design pattern 523–525
introduction to 495–509
 catalogs 504–509
 defining imports and exports 499–504
 resolving objects 496–499
managing lifetime 509–513
 declaring creation policy 509–511
 releasing objects 511–513
multiple components 513–521
 selecting among multiple candidates 513–516
 wiring 516–521
MEF Contrib open source project 505
MEF DI CONTAINER 312
MefOliveOil class 502
memory, preventing leaks of 364, 398–399
MenuRegistry class 360
MessageBox.Show method 293
messageWriter application 17
METHOD INJECTION pattern 111–117
 description of 111–112
 examples of 113–117
 related patterns 117
 when to use 112–113
MethodInfo instance 446
Microsoft Solutions Framework. *See* MSF
Microsoft, and containers 89–92
Microsoft.Practices.Unity.Configuration 456
Microsoft.Practices.Unity.InterceptionExtension assembly 485

minimum size option 267
misconceptions 5–8
 is Abstract Factory 7
 only for late binding 6
 only for unit testing 6
 requires container 7–8
Model View ViewModel (MVVM) 178, 221
Module class 428
Money class 115
MoneyContract 213
MousseAuChocolat 335, 372, 403, 405, 515, 519
MSF (Microsoft Solutions Framework) 90

N

Name property 44, 46
named components 437–438, 479–480
 configuring 372–373, 401–402, 475–476
 Decorator design pattern with 378, 406
 registering 335–336, 435–437
named exports, importing 515–516
NamedParameter 443
natural clusters 183
NDepend, monitoring coupling using 193
NeedyClass 98
.NET 4.0 88
.NET Base Class Library 123
.NET DI CONTAINERS 59
.NET types, in XML 390–391
new keyword 230
newing up, DEPENDENCIES 136–137
non-public constructors, parts with 522–523
NonShared 510
Null Object 10, 106
NullReferenceExceptions 10, 105, 121–122, 289, 408
NullService 273

O

Object Builder module 91
OBJECT COMPOSITION 199–235
 for ASP.NET applications 224–230
 for ASP.NET MVC applications 206–210
 CommerceControllerFactory example 208–210
 with custom Controller Factory 206–208
 for console applications 202–205
 for PowerShell 230–235
 for WCF applications 210–219
 extensibility for 211–212
 ProductManagementService example 212–219
 for WPF applications 219–224

- object element 387, 406
 - object graph, per request 78
 - OBJECT LIFETIME 27, 239
 - Object property 21
 - object scopes, configuring 398–399
 - objectCache field 368
 - ObjectContext class 32, 116, 151
 - ObjectDataSource control 226, 228
 - ObjectFactory class 349, 357
 - objects
 - composition of 25–26
 - lifetime of 26
 - ObservableCollections 221
 - OCP (OPEN/CLOSED PRINCIPLE) 106, 283–284
 - OfType method 400
 - OfType<T> filter 400
 - OliveOil DEPENDENCY 391, 393
 - OliveOil property 502, 505
 - OliveOilAdapter class 502–503
 - OnActivating event 446
 - OnActivating method 446
 - OnStartup method 220, 223–224
 - OPEN/CLOSED PRINCIPLE. *See* OCP
 - [OperationContract] attribute 213, 298
 - OrderAdded method 187
 - OrderRepository class 187–188, 248
 - OrderService class 185–188, 248
 - OrderService example 185–188
 - outside-in technique 43
- P**
-
- p&p (patterns & practices) 90
 - packaging, configuration 322–323, 358–361, 427–429
 - Page class 225
 - Page objects 225
 - parallel development 16, 19
 - Parameter class 446
 - Parameter Objects 183
 - ParameterInfo object 437
 - params array 426
 - [PartCreationPolicy] attribute 509–510
 - Parts property 508
 - parts, importing 503–504
 - patterns 95–132
 - AMBIENT CONTEXT 118–132
 - caching currency 123–130
 - challenges 122–123
 - description of 118–120
 - example of 123
 - implicitness 122
 - related patterns 130–132
 - when to use 120–121
 - CONSTRUCTOR INJECTION 98–103
 - description of 98–99
 - examples of 100–103
 - related patterns 103
 - when to use 99–100
 - for containers 75–87
 - COMPOSITION ROOT pattern 75–81
 - REGISTER RESOLVE RELEASE pattern 81–87
 - for INTERCEPTION 281–285
 - METHOD INJECTION 111–117
 - description of 111–112
 - examples of 113–117
 - related patterns 117
 - when to use 112–113
 - PROPERTY INJECTION 104–110
 - description of 104–105
 - examples of 107–110
 - related patterns 110
 - when to use 105–107
 - patterns & practices. *See* p&p
 - Per Dependency 429
 - PER GRAPH
 - objects, lifetime for 259–261
 - sharing repository within graph example 260–261
 - when to use 260
 - Per Lifetime Scope 429
 - performance monitoring 285
 - PerRequest 361
 - PerResolveLifetimeManager 472
 - PerWebRequest 324–325, 327
 - Plain Old CLR Objects. *See* POCOs
 - Ploeh.Samples.MenuModel namespace 457
 - plug-ins, configuring multiple implementations of same 371–372
 - POCOs (Plain Old CLR Objects) 44
 - Pointcut 411
 - policySource 470
 - pool cleanup option 267
 - pool preparation option 267
 - POOLED lifestyle 325–327
 - lifetime for 266–271
 - reusing expensive repositories example 268–271
 - when to use 267
 - PooledWithSize method 326
 - PostSharp SDK 297
 - PostTearDown method 471
 - PowerShell, OBJECT COMPOSITION for 230–235
 - PreBuildUp method 470–471
 - PresentationLogic library 190, 193, 220
 - PresentationLogicUnitTest 190–191
 - PresentationModel 191
 - primitive DEPENDENCIES 341–343, 380–381, 412–413

- primitive parts 521–522
 - [PrincipalPermission] attribute 295–296
 - PrincipalPermission class 295
 - PrincipalPermission demand 295
 - Proceed method 306, 408–409
 - Process method 356, 363
 - Product class 40–41, 46, 49–50
 - Product entity 32
 - ProductChannelFactory 222
 - ProductContract 213
 - ProductManagementAgent class 290
 - ProductManagementClient 220–224, 299
 - ProductManagementClientContainer 224
 - ProductManagementClientInterceptorSelector class 308
 - ProductManagementService class 215, 217–218
 - ProductManagementService example 212–219
 - ProductManager roles 294–295
 - ProductRepository class 47–49, 149–151, 212–213, 256–257, 279–281
 - ProductRepositoryFactory 137–139, 141
 - ProductService class 33–35, 40, 43, 46, 51, 137–139
 - using SERVICE LOCATOR anti-pattern 156–157
 - with FOREIGN DEFAULT 144–146
 - ProductViewModel 44, 46
 - ProductWcfAgent library 250, 298–299
 - Program class 203
 - programming to interfaces 15
 - PropertiesAutowired method 445
 - property element 415
 - PROPERTY INJECTION 103, 105, 149, 177–178, 383, 416
 - PROPERTY INJECTION pattern 104–110
 - description of 104–105
 - examples of 107–110
 - related patterns 110
 - when to use 105–107
 - wiring with 344–346, 382–384, 445–447, 489–491, 523–525
 - AUTO-WIRING 415–416
 - explicitly 415
 - providers, currency 101–103
 - PublicKeyToken 391
 - purpose, of DI 8–13
- R**
-
- readonly keyword 107, 244, 258
 - refactoring 162–196
 - for Constructor Over-injection 182–188
 - OrderService example 185–188
 - refactoring to Facade Services 183–184
 - for short-lived DEPENDENCIES 170–175
 - mapping runtime values to ABSTRACTIONS 163–170
 - CurrencyProvider example 168–170
 - design considerations 166
 - selecting routing algorithm example 166–168
 - monitoring coupling 188–196
 - integration-testing coupling 191–193
 - unit-testing coupling 189–191
 - using NDepend 193
 - resolving cyclic DEPENDENCIES 175–181
 - breaking cycle with PROPERTY INJECTION 177–178
 - composing Window example 178–181
 - sequences to better course 337
 - toward DI
 - BASTARD INJECTION anti-pattern 148–149
 - CONSTRAINED CONSTRUCTION anti-pattern 152–154
 - CONTROL FREAK anti-pattern 143–144
 - SERVICE LOCATOR anti-pattern 159–161
 - ReferenceMatchesDefinition method 193
 - references, instance
 - Decorator design pattern with 377–378
 - wiring 373–374
 - register element 457
 - Register method 81, 83, 318–319, 442, 444
 - Register phase 82
 - REGISTER RESOLVE RELEASE pattern, for
 - containers 81–87
 - dynamic structure for 84–85
 - static structure for 82–83
 - RegisterAssemblyTypes method 74, 425–426
 - registering
 - components
 - with code blocks 343–344, 487–489
 - with custom lifetimes 467
 - difficult APIs 442–447
 - configuring primitive DEPENDENCIES 443
 - registering objects with code blocks 444–445
 - wiring with PROPERTY INJECTION design pattern 445–447
 - multiple implementations of same service 333–334, 473–475
 - named DEPENDENCIES 335–336, 435–437
 - objects with code blocks 444–445
 - RegisterModule overloads 428
 - RegisterType method 421–424, 426, 454–455, 459–460
 - RegisterType overload 460
 - registration, automatic 355–357, 424–426, 455–456
 - Registry class 353, 358
 - RegularExpressionMethodPointcut 411
 - Release method 81–82, 86–87, 207, 243
 - Release phase 82
 - ReleaseController method 86–87, 206, 208

ReleaseExport method 512
 ReleaseInstance 211
 releasing
 components 431–433, 461–464, 467–469
 parts 511–513
 disposing container 512–513
 exports 512
 Remove-Basket cmdlet 232
 RemoveValue method 465, 467, 472
 repository DEPENDENCY 242, 258
 repository member 136, 248
 RepositoryBasketDiscountPolicy class 164, 240–241, 259, 261
 repositoryForCampaign 241
 repositoryForPolicy 241
 RequestContext 207
 requests, resolving 389
 RequiredCreationPolicy property 511
 Resolve method 317, 324, 328, 330, 332, 459
 Resolve phase 82
 ResolveAll method 334, 339, 474, 477
 ResolvedArrayParameter<ICourse> instance 478
 ResolveDiscountRepository method 264
 ResolvedParameter class 436–437, 443, 446, 476, 490
 ResolveHomeController method 260–261
 ResolveLease method 330
 ResolveNamed method 435
 ResolvePresenter method 229
 ResolveProductManagementService method 218, 243, 270
 resolving
 multiple repositories example 258–259
 named components by convention 437–438
 objects 387–389, 420–422, 451–453
 mapping ABSTRACTIONS to concrete types 316–317, 421–422, 497–498
 resolving weakly typed services 351–352, 452–453, 498–499
 type requests 389
 weakly typed services 317
 sequences 339
 weakly typed services 317, 351–352, 422
 resources
 embedded, loading XML using 395
 XML, combining 396–397
 result variable 409
 reusing expensive repositories example 268–271
 Rillettes 336, 403, 405, 515, 519
 RouteController 167
 RouteSpecification 168
 RouteType 167
 RRR. *See* REGISTER RESOLVE RELEASE
 run-time INTERCEPTION 198

runtime values, mapping to ABSTRACTIONS
 163–170
 CurrencyProvider example 168–170
 design considerations 166
 selecting routing algorithm example 166–168

S

Salutation class 14–15, 18, 22, 27
 sample Commerce application example, configuring containers for
 by convention 73–75
 with code 71–72
 with XML 69–70
 SauceBéarnaise class 316, 318, 320, 496–498
 SauceBéarnaise instance 388
 SauceCatalog 508
 SauceConvention class 356
 save the order action 185
 Scan method 355–356
 Scope attribute 364
 scope, of DI 24–28
 INTERCEPTION 26–27
 OBJECT COMPOSITION 25–26
 OBJECT LIFETIME 26
 three dimensions of 27–28
 SEAMS 22
 SecureMessageWriter class 18, 27
 SecureProductRepository Decorator 294
 security aspect 285
 security, CROSS-CUTTING CONCERNS 293–295
 SelectAllProducts method 289
 selecting
 among multiple candidates
 configuring named DEPENDENCIES 372–373, 401–402, 475–476
 importing named exports 515–516
 of same ABSTRACTION 400–401
 of same component 473–475
 of same export 514–515
 of same plug-in 371–372
 of same service 333–334, 434–435
 registering named dependencies 335–336, 435–437
 resolving named components by convention 437–438
 wiring instance references 373–374
 configuring multiple implementations of same abstraction 371–372, 400–401, 434–435
 routing algorithm example 166–168
 SelectInterceptors method 307
 SelectProduct method 279
 sequences 336–339, 374–377, 402–405
 AUTO-WIRING 374–375, 402–404
 configuring arrays 337–338

- sequences (*continued*)
 - picking exports from larger set 517–519
 - picking objects from larger set 404–405
 - refactoring to better course 337
 - resolving 339
 - wiring 438–440, 476–479, 516–519
 - AUTO-WIRING 438, 476–477, 516–517
 - picking exports from larger set 517–519
- [Serializable] attribute 265
- SERVICE LOCATOR 7, 77, 135, 154–161, 216, 226
 - analysis of 157–161
 - impact 157–159
 - refactoring toward DI 159–161
 - example of, ProductService class 156–157
- ServiceActivationException 210
- [ServiceContract] attribute 213, 298
- ServiceHost 212–213, 215, 298
- ServiceHostFactory 77, 212–215
- ServiceOverrides method 335–336, 338–342, 345
- Session Request Context objects, lifetime for 265
- Set method 368
- Setter<T> method 383
- SetValue method 465, 467
- Shared value 510
- sharing repository within graph example 260–261
- shopping baskets, adding currency providers
 - to 101–103
- ShowDialog method 181
- single extension method 499
- SINGLE RESPONSIBILITY PRINCIPLE. *See* SRP
- SingleInstance method 246, 429–430, 433
- SINGLETON LIFESTYLE 242, 255–256
- Singleton method 361–362
- Singleton objects, lifetime for 255–258
 - thread-safe in-memory repository example 256–258
 - when to use 256
- SingletonConvention 363
- Site property 107
- SlidingLease class 331–332, 370
- SmartInstance<T> class 374, 379, 381
- SmartInstance<VealCutlet> 379
- SOLID principles 283
- SomeAspect.tt 302
- SomeAspectDecorator.cs 302
- SomeClass 104, 111
- SomeLifetimeManager 464
- SomeService class 153
- SomeServiceFactory class 153
- spiciness parameter 342, 380, 413, 443
- Spiciness.Hot 342, 381, 390, 413
- Spring.NET DI CONTAINER 312, 385–416
 - configuring difficult APIs 412–416
 - primitive DEPENDENCIES 412–413
 - static factories 413–414
 - wiring with PROPERTY INJECTION design
 - pattern 414
- introduction to 386–397
 - configuring container 389–393
 - loading XML 393–397
 - resolving objects 387–389
 - managing lifetime 397–399
- multiple components 399–412
 - INTERCEPTORS 407–412
 - selecting among multiple candidates 400–402
 - wiring 402–407
- SQL Server Management Studio 32
- SQL Server-based data access 72
- SqlAuditor 284
- SqlBasketRepository 66
- SqlCampaignRepository 229
- SqlCurrencyProvider 204
- SqlDataAccess library 191
- SqlDiscountRepository instance 241, 260
- SqlOrderRepository 248
- SqlProductRepository class 137–140, 243–246
- sqlRepository 280
- SRP (SINGLE RESPONSIBILITY PRINCIPLE) 99, 183–184, 284
- srcv variable 254
- stability pattern 286
- STABLE DEPENDENCIES 23
- StartupUri attribute 219, 223
- static factories 140–142, 413–414
- static keyword 230
- static structure, for RRR pattern 82–83
- Steak class 427
- StreamReader 101
- streams, loading XML using 395–396
- StreamWriter 101
- StructureMap DI CONTAINER 71, 73, 90, 312, 347–384, 422
 - configuring difficult APIs 380–384
 - objects with code blocks 381–382
 - primitive DEPENDENCIES 380–381
 - wiring with PROPERTY INJECTION design
 - pattern 382–384
 - introduction to 348–361
 - configuring container 352–358
 - packaging configuration 358–361
 - resolving objects 350–352
 - managing lifetime 361–370
 - multiple components 370–380
 - selecting among multiple candidates 371–374
 - wiring 377
- StructureMap.IContainer interface 352
- SupplierReorderPolicy 248
- SUT (System Under Test) 19, 128, 146
- System.Attribute 296
- System.ComponentModel namespace 113

System.ComponentModel.Component 316, 318
 System.ComponentModel.Design.IDesigner 113
 System.ComponentModel.IComponent 107
 System.DateTime.Now 24
 System.IO.StreamReader 100
 System.IO.StreamWriter 100
 System.Object 63, 317
 System.Random 24
 System.Runtime.Remoting.Lifetime.ILease 330
 System.Security.Cryptography.RandomNumber-
 Generator 24
 System.Security.Permissions.PrincipalPermission
 class 294
 System.Security.Principal.IPrincipal interface 123
 System.ServiceModel.ChannelFactory<TChannel>
 175
 System.Web.HttpContext 166
 System.Web.Mvc.ViewPage 44
 System.Web.Mvc.ViewPage<FeaturedProducts-
 ViewModel> 44
 System.Windows.Window 179

T

TDD (Test-Driven Development) 6, 20, 43, 90,
 124, 190
 Teardown method 462, 471–473
 technical problem 41
 TESTABILITY 16, 19–20
 testable 19
 Test-Driven Development. *See* TDD
 Third-party Connect 76
 this.agent 174
 this.container 352, 422
 Thread Context objects, lifetime for 265–266
 Thread.CurrentPrincipal 123, 294–295, 299
 Thread.CurrentUICulture 123
 Thread.Sleep 240
 ThreadLocal 361
 thread-safe in-memory repository example
 256–258
 thread-safety 80
 Three Calls Pattern 82
 ThreeCourseMeal class 335–336, 372–374,
 515–516
 ThreeCourseMeal constructor 436
 ThreeCourseMeal objects 401–402
 three-layer application diagram 30
 tightly coupled code 9
 time, modifying 128–130
 TimeProvider class 120, 125–126, 128–129, 332
 TimeProvider parameter 120
 TimeProvider Stub 129
 TimeSpan 151, 412
 Total property 232

Trace class 123
 Trace.Listeners property 123
 Trace.Write method 123
 TRANSIENT objects, lifetime for 258–259
 resolving multiple repositories example
 258–259
 when to use 258
 TransientLifetimeManager 472
 type catalogs 505–506
 Type instance 156, 389
 type requests, resolving 351–352, 389, 452–453,
 498–499
 TypeCatalog 504–506
 TypeConverter class 113
 typeof(BasketController) 65
 types, exporting 500–501
 type-safe approach 447

U

UI-neutral 221
 Uniform Resource Identifiers. *See* URIs
 Uninterrupted Power Supply. *See* UPS
 Unique lifestyle 361, 382
 unit testing
 benefits of DI 21
 misconceptions about DI 6
 UnitPrice property 44, 46
 unit-testing coupling 189–191
 Unity DI CONTAINER 312, 448–491
 configuring difficult APIs 486–491
 primitive DEPENDENCIES 486–487
 registering components with code
 blocks 487–489
 wiring with PROPERTY INJECTION design
 pattern 489–491
 introduction to 450–459
 configuring container 453–458
 packaging configuration 458–459
 resolving objects 451–453
 managing lifetime 459–473
 configuring 460–464
 developing custom 464–473
 multiple components 473–486
 INTERCEPTORS 481–486
 selecting among multiple candidates 473–476
 wiring 476–480
 UnityContainer class 62, 451
 UnityContainer instance 467
 UnityContainerExtension class 458
 UpdateCurrency program example 202–205
 UpdateCurrencyCode method 108
 UpdateProduct method 279
 UPS (Uninterrupted Power Supply) 11
 UriResource 393

URIs (Uniform Resource Identifiers) 395
 Use method 382
 User Interface layer, for complex example
 right way 43–45
 wrong way 34–36
 using scope 432
 UsingFactoryMethod 343–345

V

Value property 497
 valueAccessor 437
 VealCutlet
 class 341
 component 440–442
 instance 377, 379, 441
 object 406
 type 406, 441
 Verify method 21
 ViewModels 179–181, 220
 ViewResult instance 35, 52
 vmFactory 181
 VOLATILE DEPENDENCIES 23–24, 136

W

WCF applications 210–219
 extensibility for 211–212
 ProductManagementService example 212–219
 wcfAgent variable 290
 WcfProductManagementAgent class 222–223,
 292–293, 298–299
 weakly typed services, resolving 351–352, 452–453,
 498–499
 WEB REQUEST CONTEXT objects, lifetime for
 261–266
 and Session Request Context objects 265
 and Thread Context objects 265–266
 HomeController with request-shared repository
 example 262–265
 when to use 261–262
 web.config 51, 70
 WillExecute property 482
 WindowAdapter 179
 Windows Azure Table Storage Service 150
 Windows Communication Foundation. *See* WCF
 Windows Complication Foundation 210
 Windows Presentation Foundation. *See* WPF
 Windsor example 303–309
 activating Interceptors 306–309
 Circuit Breaker Interceptor 305–306
 exception handling Interceptor 303–305
 WindsorContainer 63, 315, 318, 323, 341
 WindsorControllerFactory 80, 86

wiring
 automatic
 DEPENDENCIES 392–393
 sequences 402–404, 438, 516–517
 AUTO-WIRING sequences 374–375
 Decorator design pattern 339–341, 440–442,
 479–480, 519–521
 concrete component 480
 named component 479–480
 with concrete contracts 519–521
 with delegates 441–442
 with WithParameter method 440–441
 explicitly, Decorator design pattern 339–340
 implicitly, Decorator design pattern 340–341
 instance references 373–374
 picking components from larger set 438–440,
 477–479
 sequences 336–339, 438–440, 476–479, 516–519
 AUTO-WIRING 438, 476–477, 516–517
 configuring arrays 337–338
 picking exports from larger set 517–519
 refactoring to better course 337
 resolving 339
 with PROPERTY INJECTION pattern 344–346, 382–
 384, 445–447, 489–491, 523–525
 AUTO-WIRING 415–416
 explicitly wiring 415
 WithParameter method 436–437, 439–443, 445
 WithProperty method 445–446
 WithService property 320
 WorkflowRuntime 107–108
 WorkflowSchedulerService 108
 WPF (Windows Presentation Foundation) 38, 173,
 219–224
 wpfWindow field 179
 WpfWindow property 181
 Write method 18, 21
 writer instance 13
 writerMock 21

X

XamlSchemaContext 222
 XferProductRepository class 268–270
 XML (Extensible Markup Language)
 .NET types in 390–391
 configuration of 357–358, 426–427, 456–458
 configuring 321–322
 configuring containers with 68–70
 configuring instance scope with 431
 configuring lifestyles with 324–325, 363–364
 loading 393–397
 combining XML resources 396–397
 from URIs 395
 using application configuration files 394–395

XML (*continued*)

- using embedded resources 395
- using streams 395–396
- using XML files 394
- managing lifetimes with 461

XML style 68

- XmlApplicationContext class 388, 396–398
- XmlObjectFactory 396, 398
- XmlProductRepository 151
- XmlReader 151