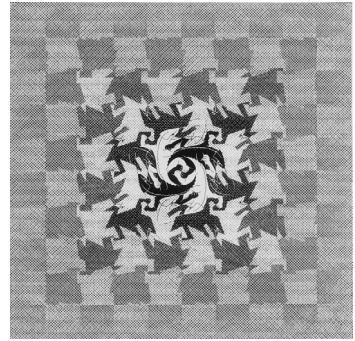


Index



A

- Abstract Cache pattern, 43, 320–342
 - ASP.NET caching, 338–342
 - Cache, 325
 - cache container:
 - adding items to, 339–341
 - retrieving cached items from, 341
 - CacheAdapter, 325
 - class diagram, 327
 - Client, 325
 - ConcreteCacheable, 322, 325
 - consequences, 322–324
 - forces, 322
 - ICache, 326
 - ICacheable, 326
 - IEnumerable, 326
 - implementation, 326–342
 - implementation sample of preparing a cache using, 333–335
 - intent, 320
 - lazy cache service implementation, 332–333
 - participants, 325–326
 - problem, 320–322
 - related patterns, 342
 - setting SOAP headers using .NET, 331
 - SOAP headers, 327–342
 - SOAP message header example, 330
 - SOAP Toolkit Visual Basic header serialization example, 328–329
 - structure, 322
 - wrapping a cache container (implementation sample), 336–338
- Abstract Packet pattern, 42, 142, 178–192, 214, 257, 260
 - benefits/liabilities, 181–183
 - boxing/unboxing, 185–184
 - consequences, 181–183
 - DataSet, 183
 - forces, 180
 - generic class diagram, 181
 - implementation, 183–192
 - implementation class diagram, 180
 - intent, 178
 - Packet, 183
 - Packet Translator, 183
 - participants, 183
 - problem, 179–180
 - ProductDataSet, 183
 - related patterns, 192
 - structure, 181
- Abstract Schema, 42, 203
- ADO DataSets, 24
- ADO.NET, 202–204, 209, 220, 232, 236, 363

- Advanced design patterns, 43, 319–401
 - Abstract Cache pattern, 320–342
 - ASP.NET caching, 338–342
 - Cache, 325
 - cache container:
 - adding items to, 339–341
 - retrieving cached items from, 341
 - CacheAdapter, 325
 - class diagram, 327
 - Client, 325
 - ConcreteCacheable, 325
 - consequences, 322–324
 - forces, 322
 - ICache, 326
 - ICacheable, 326
 - IEnumerable, 326
 - implementation, 326–342
 - implementation sample of preparing a cache using, 333–335
 - intent, 320
 - lazy cache service implementation, 332–333
 - participants, 325–326
 - problem, 320–322
 - related patterns, 342
 - setting SOAP headers using .NET, 331
 - SOAP headers, 327–342
 - SOAP message header example, 330
 - SOAP Toolkit Visual Basic header serialization example, 328–329
 - structure, 322
 - wrapping a cache container (implementation sample), 336–338
 - Loosely Coupled Transactor Client (LCT Client), 380–400
 - benefits/liabilities, 383–384
 - CLCTClientSoapSocket, 385
 - CLCTClientThreadPool, 384
 - consequences, 383–384
 - CSoapSocketClientT, 385
 - forces, 380–381
 - implementation, 385–400
 - intent, 380
 - LCT Client C++ console implementation, 385–388
 - LCT Client “plumbing” implementation, 388–391
 - LCT Client “plumbing” implementation (template code), 392–399
 - participants, 384–385
 - problem, 380
 - structure, 381
 - Loosely Coupled Transactor Server (LCT Server), 350–379
 - asynchronous system design challenges, 350–354
 - benefits/liabilities, 359–362
 - class diagram, 364
 - Client, 362
 - consequences, 359–362
 - ExternalService, 363
 - ExternalServiceProxy, 363
 - forces, 357
 - generic class diagram, 358
 - implementation, 364–379
 - intent, 350
 - ITransactor, 363
 - LCT Server “plumbing” implementation, 366–369
 - participants, 362–363
 - problem, 354–357
 - related patterns, 379
 - sample functional implementation drive by, 370–379
 - structure, 358–359
 - TransactorFacade, 363
 - TransactorService, 362–363
 - Visual Basic SOAP Toolkit client used for calling, 363
 - Password Storage pattern, 400–401
 - benefits/liabilities, 401
 - consequences, 401
 - forces, 400
 - hashing helper method implementation, 341
 - intent, 400
 - participants, 401
 - problem, 400
 - structure, 401
 - Web Service Interface pattern, 342–350
 - benefits/liabilities, 344–346
 - client-side WSI implementation example, 350
 - concrete WSI implementation example, 347–348
 - consequences, 344–346
 - forces, 343
 - implementation, 347–350
 - intent, 342
 - IWebServiceInterface, 346
 - participants, 346–347
 - problem, 342–343
 - sample interface, 347
 - ServiceClient, 346
 - ServiceProxy, 346
 - structure, 344
 - Web service piece of WSI implementation, 349

WebServiceConcrete, 346
 WebServiceInterface, 346
 WSDL-generated proxy code, 349–350
 Alexander, Christopher, 38–39
 American Clearing House Association, 274
 Application level, 76
Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design, 39
 .asmx, 20–21
 ASP.NET, 63, 278, 338–339
 Assembly, System.Reflection namespace, 160
 Asynchronous Web services, 286
 Authorize method, 170

B

Base Class Library (BCL), 9, 35
 Base exception class, 47
 BaseException base class, 49, 51

- adding SOAP exception support to, 71–77
 - COM exception handling, 75–76
 - determining when to log, 76–77
 - sample BaseException actor for allowing SOAP Faults, 72–73
 - sample SOAP Fault builder for, 71–72
 - sample SOAP Fault detail node builder, 74–75
 - using XML, 76
- applying the child class, 57
- beginning of, 54–57
- building, 53–63
- features handled by, 51–52
 - automated call stack formatting and display, 51
 - automated error stack formatting and display, 51
 - automated exception chaining, 51
 - custom remote exception handling using SOAP, 52
 - error logging and message tracing, 52
- logging, 52–53
 - environmental information, 60–61
 - error stack, building, 61–62
 - sample logging routine, 58–59
 - what to log, determining, 59–62
 - where to log, determining, 58–59
- sample child class, 50
- sample stack builder, 62
- system exceptions, throwing, 62–63

 BaseException class, 71, 288
 BooleanSwitch, 80–82
 Boundaries, 179
 Box, Don, 38
 Boxing/unboxing, 185–187
 BuildDetailNode method, 74–75
 BuildErrorStack method, 61
 Burgett, David, 298

C

C#, 3, 8, 35, 385

- delegate keyword in, 111

 CacheAdapter, 325–326, 331, 335–336
 Call stack, 61
 Caller communication, 352–353
 Categorizing patterns, 39–41
 CExternalServiceT, 384
 Chained Service Factory pattern, 42, 142, 143–152, 284

- benefits/liabilities, 146–148
- ConcreteFacade, 148
- consequences, 146–148
- Execute method, 144
- Facade, 148
- forces, 145–146
- generic class diagram, 146
- implementation, 149–152
- intent, 143
- participants, 148
- problem, 143–145
- related patterns, 152
- Service, 148
- Service Factory metadata helper method, 150–151
- Service Factory method sample implementation, 149–150
- ServiceClient, 148
- structure, 146

 CLCTClientSoapSocket, 384, 385
 CLCTClientThreadPool, 384
 Client faultcode, 66
 COM, 12, 35
 COM+, 29, 163, 166
 Commercial Framework, 267–270, 278, 304–306

- architectural features, 284–289
 - Abstract Data packet, 288
 - exception handling and tracing framework, 288
 - FTP client/FTP Web services, 286
 - instrumentation, 288
 - loosely coupled Web services, 284
 - managed client framework, 284
 - messaging and Message Listening Windows 2000 Service, 286
 - network services, 286
 - Poly model, 285
 - Product Manager, 288

- Commercial Framework (cont.)
 - remote tracing, 287
 - scheduling service, 287
 - service facades, 284
 - unmanaged client framework, 285
 - roles in, 291
 - vision statement, 282
 - Common language infrastructure (CLI), 5
 - Common language runtime (CLR), 5, 9, 29–31, 35, 75
 - Common language specification (CLS), 5
 - Common type system (CTS), 5, 36
 - ComplexReport object, 331–332
 - ComplexType element tag, 205
 - Contract interface, 144
 - CORBA, 12–13, 24
 - CreateInstance method, 161
 - CreditCardDS child class, DataSet, 196
 - Crivat, Bogdan, 380
 - CSoapSocketClientT template, 381
 - Custom SOAP Exception Handler, 41
 - Custom SOAP trace extension, 288
 - Custom trace listener, 80, 287
- D**
- Data Access Interface, 285
 - DataSet, 65, 94, 182–186, 222, 230, 313, 379
 - DataSet Visual Studio, 207–208
 - DBGetTransactionBySQL method, 259
 - DCOM, 12–13, 24
 - Decision schema, 290
 - DefaultTraceListener, 78
 - #define TRACE, 77
 - DeleteKeys method, 247
 - Design Patterns: Elements of Reusable Object-Oriented Software*, 39
 - DIME protocol (Microsoft), 24
 - Disconnected coordination/administration, 352–354
 - Distributed applications, 6
 - Distributed garbage collection, 24
 - Distributed technologies, 11–13
 - DLL-hell, 7
 - Document type definition (DTD), 26
 - DOM, 23
 - Dump method, 58–59
 - Dynamic Assembly Loader pattern, 42, 132–134
 - benefits/liabilities, 133
 - consequences, 133
 - constructing an object using Reflection, 134
 - forces, 133
 - implementation, 134
 - intent, 132
 - participants, 134
 - problem, 132–133
 - structure, 133
- E**
- EInvoke method, 306–308
 - eBusiness software, and communications management, 7
 - Edit Relation Dialog Box, constraints available in, 216
 - Eiffel, 8
 - Electronic Payments Network, 274
 - Enterprise Java Beans, 12
 - Error Cross-Reference Generator pattern, 42, 125–128
 - benefits/liabilities, 127
 - calling, 128
 - consequences, 127
 - forces, 126
 - implementation, 127–128
 - intent, 125
 - participants, 127
 - problem, 125–126
 - related patterns, 128
 - structure, 126
 - ErrorCode set, 75
 - Essential COM (Box), 38
 - Etier3.LazyCacheService class, 330
 - Event log listener, sample for adding to a global collection, 79–80
 - EventInfo, System.Reflection namespace, 160
 - EventLogTraceListener, 78
 - Exception boundaries, 60, 76
 - managing, 63–65
 - Exception chaining, 61
 - Exception wrapping, 63
 - Exceptions, throwing from Web services, 63–65
 - ExceptionHandler.asmx, 68–71
 - Extensible Markup Language (XML) web services, 5–6
 - ExternalCallback method, 359
 - ExtractOriginalData method, 379
- F**
- Faultcode, 70
 - Faultfactory property, 70
 - Federal Reserve, 274
 - automated check mechanism, 272
 - FieldInfo, System.Reflection namespace, 160
 - Filter Builder, 285
 - FinancialServiceFactory.Execute, 144
 - Framework patterns, 41, 45–105
 - application-specific exceptions, 47–53
 - BaseException base class, building, 53–63
 - custom trace listener, building, 82–104

- exception boundaries, managing, 63–65
 - exception handling, 47–53
 - overview, 45–47
 - remote tracing, 82–104
 - remote trace receiver, building, 86–87
 - remote trace viewer, building, 94–104
 - sample business object to be placed on queue, 88–92
 - sample Remote Trace Listener viewer (GUI), 94–103
 - sample RemoteTraceListener Web service, 86–87
 - sample routine for constructing/adding custom listener, 86
 - sample socket routine for sending a message, 92–94
 - sending races to message queue, 87–92
 - sending traces via sockets, 92–94
 - Trace Listener template, 84–85
 - SOAP Faults, 65–77
 - adding SOAP exception support to BaseException class, 71–77
 - detail element, 67, 71
 - faultactor, 66
 - faultcode, 66
 - faultstring, 66
 - sample SOAP Fault detail block, 67–68
 - throwing custom SOAP exceptions, 68–71
 - trace listeners, 77–80
 - FTP, 25, 316
 - FtpException class, 49–51, 54, 57, 61
 - FTPMessages string table, 50
 - FTPWebRequest, 286
 - FTPWebResponse, 286
 - Function-specific exception classes, 49
- G**
- Gang of four (GoF), 8, 38
 - General network libraries, 286
 - General Responsibility Assignment Software Patterns (GRASP), 39
 - GetDbType method, 310
 - GetPlaceList, 16–17
 - GetSchemaTemplate method, 26–27, 306, 313
 - GetTransaction method, 262, 362–363, 379, 384, 391
- H**
- Headers, SOAP, 28
 - HTTP, 25
 - HTTP GET request, 24
 - HydrateDataObject method, 194, 195, 198
- I**
- ICacheable interface, 335–336
 - ICached interface, 332
 - IErrorInfo interface, 75
 - COM, 75
 - IHeaderHandler interface, 328, 330
 - Information, and problem solving, 46
 - InitTraceListeners method, 79–80
 - Inner exception, 61
 - Intellisense, 208, 357
 - Internet protocols, 12
 - InvokeTimeConsumingMethod method, 369–370
 - IOException, 63
 - IService interface, 347
 - IWebServiceInterface, 346
- J**
- J#, 8, 35
 - JBuilder, 13
- K**
- Kumar, Pranish, 380
- L**
- Larman, Craig, 39
 - LBInvoke method, 306–308
 - LCT Client, 43
 - LCT Client C++ console implementation, 385–388
 - LCT Client “plumbing” implementation, 388–391
 - LCT Client “plumbing” implementation (template code), 392–399
 - LCT Server, 43
 - Logging, 76
 - Loosely Coupled Transactor Client (LCT Client), 380–400
 - benefits/liabilities, 383–384
 - CLCTClientSoapSocket, 385
 - CLCTClientThreadPool, 384
 - consequences, 383–384
 - CSOapSocketClientT, 385
 - forces, 380–381
 - implementation, 385–400
 - intent, 380
 - LCT Client C++ console implementation, 385–388
 - LCT Client “plumbing” implementation, 388–391
 - LCT Client “plumbing” implementation (template code), 392–399
 - participants, 384–385
 - problem, 380
 - structure, 381
 - Loosely Coupled Transactor (LTCT), 43

Loosely Coupled Transactor Server (LCT Server)
 pattern, 350–379
 asynchronous system design challenges, 350–354
 benefits/liabilities, 359–362
 class diagram, 364
 Client, 362
 consequences, 359–362
 ExternalService, 363
 ExternalServiceProxy, 363
 forces, 357
 generic class diagram, 358
 implementation, 364–379
 intent, 350
 ITransactor, 363
 LCT Server “plumbing” implementation, 366–369
 participants, 362–363
 problem, 354–357
 related patterns, 379
 sample functional implementation drive by, 370–379
 structure, 358–359
 TransactorFacade, 363
 TransactorService, 362–363
 Visual Basic SOAP Toolkit client used for calling, 363

M

Macropattern, 202
 Managed C++, 8, 35
 Message batching, and SOAP, 24
 Message routing service, 286
 MessageInfo property, 92
 Messaging Manager, 286
 MethodInfo, System.Reflection namespace, 160
 Microsoft Bizataalk, 300
 Microsoft CRM, 276–277
 Microsoft Message Queue Server, 350–351, 359, 361
 Microsoft Message Queuing, 88
 Middle-tier patterns, 42, 142–200
 Abstract Packet pattern, 42, 142, 178–192
 benefits/liabilities, 181–183
 boxing/unboxing, 185–184
 consequences, 181–183
 DataSet, 183
 forces, 180
 generic class diagram, 181
 implementation, 183–192
 implementation class diagram, 180
 intent, 178
 Packet, 183

Packet Translator, 183
 participants, 183
 problem, 179–180
 ProductDataSet, 183
 related patterns, 192
 structure, 181
 Chained Service Factory pattern, 42, 142, 143–152
 benefits/liabilities, 146–148
 ConcreteFacade, 148
 consequences, 146–148
 Execute method, 144
 Facade, 148
 forces, 145–146
 generic class diagram, 146
 implementation, 149–152
 intent, 143
 participants, 148
 problem, 143–145
 related patterns, 152
 Service, 148
 Service Factory metadata helper method, 150–151
 Service Factory method sample implementation, 149–150
 ServiceClient, 148
 structure, 146
 overview, 141–142
 Packet Translator pattern, 42, 142, 192–200
 benefits/liabilities, 194
 ConcreteFacade, 194
 consequences, 194
 DataSet, 196
 forces, 193
 generic class diagram, 193
 implementation, 196–200
 implementation class diagram, 197
 intent, 192
 packet, 196
 PacketTranslator (participant), 195
 participants, 194–196
 problem, 192–193
 ProductDataSet, 196
 related patterns, 200
 sample implementation preparing packets, 197–198
 sample implementation translating packets, 199–200
 structure, 193
 Product Manager pattern, 42, 142, 163–171
 benefits/liabilities, 166–167
 ConcreteDelegate, 168
 ConcreteProduct, 168
 consequences, 166–167

- delegation implementation, 169–170
 - Facade, 167
 - forces, 165
 - generic class diagram, 166
 - implementation, 168–171
 - intent, 163
 - ManagedProduct, 168
 - method implementation, 169
 - participants, 167–168
 - problem, 163–165
 - ProductManager, 167–168
 - related patterns, 171
 - structure, 166
 - UnmanagedProduct, 168
 - worker implementation, 170
 - Service Facade pattern, 42, 142, 171–178
 - benefits/liabilities, 174
 - ConcreteFacade, 175
 - consequences, 174
 - Facade, 175
 - forces, 173
 - generic class diagram, 173
 - implementation, 175–178
 - intent, 171
 - participants, 174–175
 - problem, 171–173
 - related patterns, 178
 - sample implementation, 176–177
 - Service, 174–175
 - structure, 173
 - Unchained Service Factory pattern, 42, 142, 152–162
 - Activator, 157
 - Assembly, 157
 - benefits/liabilities, 146–148
 - ConcreteFacade, 157
 - consequences, 155–156
 - Facade, 157
 - forces, 154–155
 - generic class diagram, 155
 - implementation, 157–162
 - implementation using Reflection, 158–158
 - intent, 152
 - MethodInfo, 157
 - .NET Reflection services, 159–162
 - participants, 157
 - problem, 152–154
 - related patterns, 162
 - Service, 157
 - ServiceClient, 157
 - structure, 155
 - Type, 157
 - Module, System.Reflection namespace, 160
 - Multisync Thread Manager pattern, 42, 122–125
 - benefits/liabilities, 124
 - consequences, 124
 - DoExecution method, 125–126
 - forces, 123
 - implementation, 124–125
 - intent, 122
 - participants, 124
 - problem, 122–123
 - related patterns, 125
 - structure, 123
- ## N
- NACHA (National Automated Clearing House Association), 270–272, 277, 280
 - Nested relationships, 212–213
 - .NET:
 - components of, 34–37
 - controllable areas in, 30–31
 - highlights of, 33–34
 - interoperability support, 35
 - base class library (BCL), 35
 - common language runtime (CLR), 9, 29–31, 35, 75
 - common type system (CTS), 36
 - full Web service and SOAP support, 36
 - object-oriented ASP.NET, 36
 - simplified deployment, 36
 - XML at core, 36
 - libraries, 4
 - and Microsoft, 43
 - overview of, 34–36
 - securing Web services in, 29–30
 - and XML Web services, 13–18
 - .NET CLR, 9, 29–31, 343
 - .NET framework, 268, 281
 - and a distributed new world, 5–13
 - major components of, 33–34
 - and object-oriented languages, 6, 8–11
 - and objects, 18
 - and OO, 8–11
 - and Web services, 18
 - .NET Framework Configuration snap-in, 30
 - .NET Reflection services, 61, 159–162
 - .NET System.Messaging library, 88
 - Notifying Thread Manager pattern, 42, 108–116
 - benefits/liabilities, 110
 - calling Notifying Thread Manager, 113
 - Client, 109–110
 - consequences, 110
 - delegate definitions, 111
 - DoExecution method notification, 114–115
 - ExecuteAsync method, 112
 - forces, 109

Notifying Thread Manager pattern (cont.)

- implementation, 111–115
- intent, 108
- Notifying Thread Manager, 109–111
- participants, 110–111
- problem, 108–109
- related patterns, 116
- structure, 109–110

O

- Object references, 24
- Object-oriented ASP.NET, 36
- Object-oriented languages, 33
 - learning, 10
 - and .NET framework, 8–11
- Originating Depository Financial Institution (ODFI), 275
- Originator, automated check transaction, 275
- OutputDebugString API, 78

P

- Packet Translator pattern, 42, 142, 192–200
 - benefits/liabilities, 194
 - ConcreteFacade, 194
 - consequences, 194
 - DataSet, 196
 - forces, 193
 - generic class diagram, 193
 - implementation, 196–200
 - implementation class diagram, 197
 - intent, 192
 - packet, 196
 - PacketTranslator (participant), 195
 - participants, 194–196
 - problem, 192–193
 - ProductDataSet, 196
 - related patterns, 200
 - sample implementation preparing packets, 197–198
 - sample implementation translating packets, 199–200
 - structure, 193
- Packet.Action property 298:
- ParameterInfo, System.Reflection namespace, 160
- Password Storage pattern, 400–401
 - benefits/liabilities, 401
 - consequences, 401
 - forces, 400
 - hashing helper method implementation, 341
 - intent, 400
 - participants, 401
 - problem, 400
 - structure, 401

Patterns:

- categorizing, 39–41
- classifying, 342
- defined, 10, 37, 37–38
- gang of four (GoF), 8, 38
- pattern library, 41–43
 - advanced patterns, 43, 319–401
 - general framework patterns, 41, 45–105
 - how to use, 43
 - middle-tier patterns, 42
 - persistence-tier patterns, 42, 201–264
 - presentation-tier patterns, 42, 107–139
- repeatable, 37
- software design patterns, 38
- usefulness of, 34
- Persistence-tier patterns, 42, 201–264
 - Abstract Schema, 42, 203
 - columns, 215
 - keys, 215–217
 - keyrefs, 216–217
 - primary keys, 215
 - unique keys, 215–216
- Poly Model Factory, 203
- Poly Model pattern, 42, 203, 217–231
 - benefits/liabilities, 223–226
 - consequences, 223–226
 - forces, 218–219
 - implementation, 226–231
 - intent, 217
 - participants, 226
 - problem, 217–218
 - related patterns, 231
 - stored procedure used for saving to DATALOG table, 229–230
 - stored procedure used for saving XML instance data to main DATA table, 230
 - StoreTransaction method, 226–228
 - storing instance data passed using DataSet/ passed-in stored procedure, 228–229
 - structure, 219–223
- Schema Field pattern, 42, 203, 231–245
 - benefits/liabilities, 235–236
 - Client, 226
 - consequences, 235
 - Data Access Object, 226
 - forces, 233
 - implementation, 235–245
 - intent, 231
 - problem, 232–233
 - related patterns, 245
 - structure, 234–235
- Schema Indexer pattern, 42, 203, 245–264
 - benefits/liabilities, 250–251

- cleaning up lookup keys from INDEX table, 253–254
- consequences, 250–251
- database-specific helper method for
 - GetTransaction, 263
- forces, 247
- helper method used to GetTransaction in Poly Model, 260–261
- helper method to store lookup key, 254–256
- implementation, 251–264
- intent, 245
- method called to save lookup column, 252–253
- method using transaction ID directly, 262
- problem, 246–247
- related patterns, 264
- sample ad hoc query method used for testing queries, 258–259
- sample schema generator and sample LOOKUP instance data for convenience, 257–258
- sample transaction retrieval method using WhereCollection SQL Builder, 259–260
- simple GetTransaction method using transaction ID directly, 261–262
- stored procedure source for storing lookup values to INDEX table, 256–257
- structure, 247–250
- schemas:
 - and DataSets, 206–208
 - nested relationships, 212–213
 - one-to-many relationships, 213
 - types of, 212–213
 - XML schema, 204–206
- table, 213–214
- typed DataSets, creating, 208–212
- Pollable Thread Manager pattern, 42, 117–122
 - AsyncReturn, 118, 119–120
 - BeginExecution method, 119
 - benefits/liabilities, 117–118
 - Client, 118
 - consequences, 117–118
 - forces, 116
 - implementation, 118–122
 - intent, 116
 - participants, 118
 - PollableThreadManager, 118–119
 - problem, 116
 - related patterns, 122
 - structure, 117
 - WaitForCompletion method, 121–122
- Poly Model pattern, 42, 202, 258, 263, 281, 283
- Poly schemas, 224
- PolyData class, 310
- PrepareCache factory method, 326, 333, 335
- PrepareConfigData method, 335–336
- PreparePacket method, 196, 302, 313
- PrepareWorkers method, 335
- Presentation-tier patterns, 42, 107–139
 - Dynamic Assembly Loader pattern, 42, 132–134
 - benefits/liabilities, 133
 - consequences, 133
 - constructing an object using Reflection, 134
 - forces, 133
 - implementation, 134
 - intent, 132
 - participants, 134
 - problem, 132–133
 - structure, 133
- Error Cross-Reference Generator pattern, 42, 125–128
 - benefits/liabilities, 127
 - calling, 128
 - consequences, 127
 - forces, 126
 - implementation, 127–128
 - intent, 125
 - participants, 127
 - problem, 125–126
 - related patterns, 128
 - structure, 126
- Multisync Thread Manager pattern, 42, 122–125
 - benefits/liabilities, 124
 - consequences, 124
 - DoExecution method, 125–126
 - forces, 123
 - implementation, 124–125
 - intent, 122
 - participants, 124
 - problem, 122–123
 - related patterns, 125
 - structure, 123
- Notifying Thread Manager pattern, 42, 108–116
 - benefits/liabilities, 110
 - calling Notifying Thread Manager, 113
 - Client, 109–110
 - consequences, 110
 - delegate definitions, 111
 - DoExecution method notification, 114–115
 - ExecuteAsync method, 112
 - forces, 109
 - implementation, 111–115
 - intent, 108
 - NotifyingThread Manager, 109–111
 - participants, 110–111
 - problem, 108–109

- Presentation-tier patterns (cont.)
 - related patterns, 116
 - structure, 109–110
 - overview, 107–108
 - Pollable Thread Manager pattern, 42, 117–122
 - AsyncReturn, 118, 119–120
 - BeginExecution method, 119
 - benefits/liabilities, 117–118
 - Client, 118
 - consequences, 117–118
 - forces, 116
 - implementation, 118–122
 - intent, 116
 - participants, 118
 - PollableThreadManager, 118, 119
 - problem, 116
 - related patterns, 122
 - structure, 117
 - WaitForCompletion method, 121–122
 - Stunt Driver pattern, 42, 135–139
 - adding items to test parameters, 139
 - benefits/liabilities, 136
 - building test set the easy way with, 138
 - consequences, 136
 - forces, 135
 - getting test parameters from, 136
 - implementation, 136–139
 - intent, 135
 - ITestable component, 136
 - participants, 136
 - problem, 135
 - related patterns, 139
 - structure, 135
 - Test Client, 136
 - testing Stunt Driver, 138
 - WebForm Template pattern, 42, 128–132
 - benefits/liabilities, 129–130
 - consequences, 129–130
 - default sample page, 132
 - forces, 128
 - implementation, 130–132
 - intent, 128
 - participants, 130
 - problem, 128
 - related patterns, 131–132
 - rendering (complete), 131
 - rendering (simple form), 130–131
 - structure, 129
 - Process patterns, 265–318
 - applying.NET patterns, 291–313
 - Product Manager pattern, 296–303
 - Service Facade pattern, 292–296
 - Unchained Service Factory pattern, 303–313
 - Commercial Framework, 267–270
 - architectural features, 284–289
 - vision statement, 282
 - .NET technology, as competitive advantage, 279–291
 - ProductX, 270–271
 - Product Manager pattern, 42, 142, 163–171, 296–303
 - benefits/liabilities, 166–167
 - ConcreteDelegate, 168
 - ConcreteProduct, 168
 - consequences, 166–167
 - delegation implementation, 169–170
 - Facade, 167
 - forces, 165
 - generic class diagram, 166
 - implementation, 168–171
 - intent, 163
 - ManagedProduct, 168
 - method implementation, 169
 - participants, 167–168
 - problem, 163–165
 - ProductManager, 167–168
 - related patterns, 171
 - structure, 166
 - UnmanagedProduct, 168
 - worker implementation, 170
 - ProductX, 94, 270–271
 - automated check payments, 273–275
 - checks vs. credit cards, 273
 - and consumers/businesses, 271–272
 - electronic check Web servicing, 275–279
 - Electronic Funds Transfer (EFT)
 - invoking framework from ProductX Web client, 313–317
 - summarizing, 317–318
 - Proof-of-concept (POC) delivery vehicles, 267
 - PropertyInfo, System.Reflection namespace, 160
 - Protocols, 12
 - Public Web services, 14–16
- R**
- Random tracing, 76–77
 - Real-time logging, 53
 - Receiver, 275
 - Receiving Depository Financial Institution (RDFI), 275
 - Reference types, 185–186
 - Referential integrity, 216
 - Reflection, 134, 156, 281
 - Activator class, 158, 161
 - Reflection services, 61
 - Reliability protocols, 24
 - Remote method invocation (RMI) protocol, 12

- Remote procedure calls (RPCs), 12, 144
 - Remote trace errors, 94
 - Remote Tracer, 41
 - Remote tracing, 82–104
 - remote trace receiver, building, 86–87
 - remote trace viewer, building, 94–104
 - sample business object to be placed on queue, 88–92
 - sample Remote Trace Listener viewer (GUI), 94–103
 - sample RemoteTraceListener Web service, 86–87
 - sample routine for constructing/adding custom listener, 86
 - sample socket routine for sending a message, 92–94
 - sending races to message queue, 87–92
 - sending traces via sockets, 92–94
 - Trace Listener template, 84–85
 - RemoteService.RemoteTrace, 85
 - RemoteTrace class, 80
 - RemoteTraceService.RemoteTrace, 86
 - RemoteTraceViewer, 94
 - RemoveAllKeys method, 253
 - ReportingWebService report, 326
 - Rootcause child node, 75
- S**
- SaveTransaction method, 301
 - Schedule Action Router, 287
 - Schedule Viewer, 287
 - Schema Field, 42
 - Schema Indexer, 42
 - Schema Indexer pattern, 258, 263
 - Schemas:
 - and DataSets, 206–208
 - nested relationships, 212–213
 - one-to-many relationships, 213
 - types of, 212–213
 - XML schemas, 204–206
 - Server faultcode, 66
 - Service Facade, 42
 - Service Facade pattern, 42, 142, 171–178, 292–296
 - benefits/liabilities, 174
 - ConcreteFacade, 175
 - consequences, 174
 - Facade, 175
 - forces, 173
 - generic class diagram, 173
 - implementation, 175–178
 - intent, 171
 - participants, 174–175
 - PaymentFacade, 293–294
 - problem, 171–173
 - related patterns, 178
 - sample implementation, 176–177
 - Service, 174–175
 - structure, 173
 - Service interface, 292
 - Service-oriented architecture, 143
 - SimpleType element tag, 205
 - Simplified deployment, 36
 - SMTP, 25
 - SOAP, 9, 174
 - Body tag, 27
 - defined, 20
 - Envelope tag, 27
 - headers, 28
 - security in, 28–31
 - success of, 26
 - Web services, 23–28
 - and XML, 24
 - and XML Web services, 24
 - SOAP Exception Fault Builder, 67
 - SOAP exception handling, 288
 - Soap Fault Builder, 41
 - SOAP Fault elements, 65, 74
 - SOAP faultcodes, 66
 - SOAP Faults, 65, 65–67
 - SOAP message, 26–27
 - sample captured using SOAP Trace Utility, 27
 - SOAP Toolkit, 65
 - SOAP Toolkit (Microsoft), 21–22, 65
 - SoapException class, 64, 68, 71, 73–74
 - SoapException.DetailElementName type, 75
 - SOAP-formatted messages, XML tags essential for understanding, 25–26
 - SoapSerializer object, 328
 - SoapServer object (SOAP Toolkit 2.0), 67–68
 - Sound designs, 40
 - SpDNPatternsUpdateschema, 240
 - Sproxy.exe, 21
 - SPROXY.EXE, 381, 384
 - SsessionId variable, 330
 - Stack builder, 61
 - State management, 352
 - StoreKeys method, 246–247, 253–254
 - StoreTransaction method, 311
 - StoreTransactionRecord method, 311–312
 - Structured exception handling (SEH), 47
 - Stunt Driver pattern, 42, 135–139
 - adding items to test parameters, 139
 - benefits/liabilities, 136
 - building test set the easy way with, 138
 - consequences, 136
 - forces, 135
 - getting test parameters from, 136

Stunt Driver pattern (cont.)
 implementation, 136–139
 intent, 135
 participants, 136
 problem, 135
 related patterns, 139
 structure, 135
 testing Stunt Driver, 138
System exception classes, 62–63
System Exception Wrapper, 41
System exceptions, 47–48
 throwing, 62–63
System.ApplicationException class, 51
System.Data.OracleClient, 312
System.Diagnostics.Switch, 80–81
System.Diagnostics.TraceListener class, 77–78
System.Exception class, 47, 61
System.Reflection namespace, members of, 160
System.Resources.ResourceManager, 50–51
System.Web.Caching.Cache object, 326, 338
System.Windows.Forms.DataGrid controls, 94
System.Xml.XmlDocument class, 75

T

T+3 settlement period, 273
tblimp.exe, 168, 170
TerraService (Microsoft), 14–15
 main page, 14
 Web methods list form, 15
TextWriterListener, 78
TIBCO, 350–351, 361
TModels (Type Models), 32
Trace listeners, 77–80
 customer, building, 82–104
Trace option, enabling, 77
Trace originator, 92
Trace receiver, 86
Trace switches, 80–82
Trace switching, 287
Trace Viewer, 287
TraceSwitch, 58, 80–82
Trace.WriteLineIf method, 58
Trace.Write/Trace.WriteLine, 78, 80
Tracing, 53–54
TransactorService, 379
TransactorService.Execute, 362
TransactorService.GetService, 362
Translate method, 183

U

UDDI, tModels (Type Models), 32
Unboxing, 186, 186–187

Unchained Service Factory pattern, 42, 142, 152–162, 284, 303–313
 Activator, 157
 Assembly, 157
 benefits/liabilities, 146–148
 ConcreteFacade, 157
 consequences, 155–156
 Facade, 157
 forces, 154–155
 generic class diagram, 155
 implementation, 157–162
 implementation using Reflection, 158–158
 intent, 152
 MethodInfo, 157
 .NET Reflection services, 159–162
 participants, 157
 problem, 152–154
 related patterns, 162
 Service, 157
 ServiceClient, 157
 structure, 155
 Type, 157
Uniform Resource Identifier (URI), 66
Universal Discovery Description and Integration (UDDI), 20, 32–33
Universal Resource Identifiers (URIs), 25
Universal Resource Names (URNs), 25
Unrecoverable exception, 47
“Using Data Caching Techniques to Boost Performance and Ensure Synchronization” (Burgett), 298
Utilities.SendSocketStream method, 92

V

ValidatePacket method, 301
Value types, 185
VB.NET, 4, 8, 35, 385
VersionMismatch faultcode, 66
Visa, 274
Visual Basic/Visual Basic C++, 12
Visual Studio .NET, 13, 33–34, 168, 171, 219, 268, 304, 354, 357, 359, 370
 sample Web service WSDL via, 21

W

Web Service, 287
 code, 63
 consumers, issues faced by, 17–18
Web Service Interface pattern, 42, 342–350
 benefits/liabilities, 344–346
 client-side WSI implementation example, 350
 concrete WSI implementation example, 347–348

- consequences, 344–346
 - forces, 343
 - implementation, 347–350
 - intent, 342
 - IWebServiceInterface, 346
 - participants, 346–347
 - problem, 342–343
 - sample interface, 347
 - ServiceClient, 346
 - ServiceProxy, 346
 - structure, 344
 - Web service piece of WSI implementation, 349
 - WebServiceConcrete, 346
 - WebServiceInterface, 346
 - WSDL-generated proxy code, 349–350
 - Web service provider, issues faced buying, 18
 - Web Service Proxy, 285
 - Web services, 281
 - Web Services Description Language (WSDL), 7
 - Web.config, adding a listener using:
 - WebForm Template pattern, 42, 128–132
 - benefits/liabilities, 129–130
 - consequences, 129–130
 - default sample page, 132
 - forces, 128
 - implementation, 130–132
 - intent, 128
 - participants, 130
 - problem, 128
 - related patterns, 131–132
 - rendering (complete), 131
 - rendering (simple form), 130–131
 - structure, 129
 - WebServiceInterface abstract class, 346
 - Windows event log, 58
 - Windows Event Viewer, 76
 - WSDL, 7, 9, 31–32
 - defined, 20
 - specification, 23
 - WS-Inspection Language, 33
- X**
- XML, 236
 - XML namespaces, 25
 - XML schemas, 26, 204–206, 281, 283, 290
 - schema element, 204–205
 - XML Web services, 9, 354
 - defined, 19–23
 - and .NET, 13–18
 - primer, 19–32
 - XML-formatted system configuration files, 29
 - XmlNode class, 75
 - .XSD, 208–209