

## Numerics

101 LINQ Examples 237

## A

abandon-ship strategy 179

  example 192

  when suitable 184

*See also* versioning

abstract syntax trees 8, 109, 206, 239

  printing 111

AbstractAstAttribute 127

AbstractAstMacro 118, 260

abstraction 5, 40, 67, 257

AbstractTransformerCompilerStep class 131

actions 13, 19, 40, 50, 74, 106

  commands, and 229

  language reference best

  practices 227

Actipro

  SyntaxHighlighter 198

adapter strategy 182

*See also* versioning

adapters 67

AddFileNameProperty 278

additive-change

  strategy 180–182, 184

  backward compatibility 180

*See also* versioning

administrators 76, 91, 105

after actions 255

Aho, Alfred 8

algorithms 3

analysts 46

anonymous base class 154

anonymous blocks 30, 58, 81, 112

ANTLR 8, 283

Apache Ant 44

APIs 51

  API surface area 155

  design 47

  public APIs 176

AppDomain 98–100, 135, 241, 243

  sandboxed

    AppDomain 100

  application code 65, 68

  application design 265

  application integration 87

  perspective 89

  application services 149

  application-configuration

    DSLs 89

  application-testing DSL 171

AqiStar.TextBox 199

arrays 71, 293–294

  creating 295

  slicing 297

ASP.NET 70, 147

*See also* .NET 70

aspect-oriented

  programming 25

ASPX files 147

assemblies 95, 98, 110, 123, 297

  in-memory assemblies 98

  assembly leakage 98

  assembly leaks 243

  assembly level 2

  assert method 110

ASTs 8, 109, 138, 213, 249

  attributes 18, 25, 109, 113, 126

  compiler

    transformations 253

  diagrams 113

  expression 111–112

  generating AST code

    dynamically 114

  macros 18, 109, 113, 115, 117, 138

  manipulations 115, 239

  nodes 110, 125, 259

  parsing 109

  parsing textual DS into 152

  scanning 109

  similarities with LINQ 251

  substitution 262

  transformations 251

  transforming 109

  use of processing at

    runtime 251

  views 251

*See also* abstract syntax trees

asynchronous messages 66

audit

  information 106, 276

  process 249

  trail 245

auditable actions 253

auditing 244, 276

authentication module 105

authorization 65, 72, 245

  rules 72–73, 78, 91, 158

  script 76

  system 77–78, 91

Authorization class 73–74

- Authorization DSL 63, 80, 84, 89, 143
    - Authorization class 158
    - AuthorizationRules.xml files 159
    - BooParser 214
    - callback to the Authorization gateway 148
    - class diagram 156
    - creating 72
    - dark side of 78
    - design of 72
    - DSL writer 217
    - engine 75
    - implementation of 73, 76
    - Operation property 92
    - requirements 73
    - rules, UI representation 216
    - specification 73
    - test case 159
    - testing 155–156, 158
    - tests for engine error handling 159
    - visualization 213
  - AuthorizationDsEngine 76
  - AuthorizationRule class 74
  - AuthorizationRulesParser 215
  - AutoCAD 16
  - auto-import compiler step 144
  - AutoImportCompilerStep 144
  - automatic DSLs 16
  - automatic typecasting 24
  - automatic variable declaration 23
  - AutoReferenceFilesCompilerStep 138, 144
- B**
- 
- backend processing systems 64
  - background compilation 244
  - Backus-Naur Form 8, 25, 176
  - backward compatibility 180
  - Bake 19
    - Bake DSL 44
  - Base Class Library 28
  - base class moniker 68
  - base language 6
  - batch compilation 134, 136, 146
  - BDD. *See* behavior-driven development
  - behavior-driven development 20
  - behavior-driven specifications 7
  - behaviors, documenting implementation 234
  - Benatti, Georges 19
  - best practices 257
  - Binsor 14, 190, 282
    - configuration file 45
    - versioning 190
  - BizTalk orchestrations 9
  - BNF. *See* Backus-Naur Form
  - Boo 12, 94
    - AbstractAstMacro 117
    - adding keywords 201
    - additional information 285
    - advanced features 285
    - array literal 294
    - arrays 294
    - assert statement 110
    - AST 115, 251
      - attribute, built-in 128
      - attributes 126–127
      - attributes, using with a DSL 128
      - manipulation 239
      - modifications 239
    - basic reference 285
    - basic syntax 25
    - behavior 285
    - benefits 37
    - blocks 307
    - Boo 0.9.2 111
    - Boo DSL 17
    - Boo editor 199–200
      - .boo extension 286
    - Boo syntax shortcuts 29
    - Boolean expressions 287
    - Boolean values 287
    - Brail 190
    - building blocks 312
    - building simple DSLs 262
    - built-in macros 312
    - class declaration 302
    - CLR classes 33
    - CLR language 17, 293
    - CLR-based language 23
  - CodeDOM 216
  - comments 288
  - commercial use 22
  - comparison with C# 29, 38
  - compiler 18, 69
    - extensibility 24
    - pipeline 109, 128, 131
    - pipeline, compiling with a custom pipeline 131
    - steps 128
    - transformations 111–112
  - conditionals 303
    - comparison with C# 304
  - control statements 288
  - correlated macros 125
  - custom keywords 164
  - custom parsing 282
  - date keyword 76
  - defines 258
  - definition 16
  - description of 22
  - #develop 199
  - DSL to UI 215
  - DSLs 239
    - enumeration 307
    - errors 104
    - explicit types 297
    - extending 52
    - extending parser 283
    - extensibility
      - mechanisms 132
    - extensible architecture 129
    - extension mechanism 311
    - for statement 292
    - generators 300
    - generic arguments 273
    - global namespace
      - import 144
    - hashes 295
    - host language for DSLs 18
    - imports 299
    - inheritance 306
    - interactive shell 285
    - internal DSLs 16
    - interpreter 285
    - iterations 305
    - Java 17
    - keywords 30, 302
    - language definition 200
    - language reference 302
    - language syntax 302

- Boo (*continued*)
  - language-oriented
    - features 29
  - lexical info 120
  - lists 293
  - locking objects 312
  - loops 305
  - macros 76, 112, 115, 167, 274, 312
    - comparison with C# 312
    - correlated 125
    - nested 124
    - reasons to prefer over
      - meta-methods 123
  - MacroStatement 118
  - meta-methods 246
  - method declarations 307
  - method definitions 297
  - method return type 307
  - method syntax comparison
    - with C# 307
  - methods 298, 307
  - multilevel if statements 304
  - nested macros 124
  - null 304
  - objects 298
  - of keyword 270, 273
  - operators 287
  - overview 22
  - parameter types 297, 308
  - parameters 307
  - parser 116, 259, 283
  - parsing 206
  - processing macros 124
  - programming language 285
  - programs 285
  - Python-inspired syntax 22
  - quasi quotations 113, 116
  - range 294
  - real programs 297
  - relaxed symbols 145
  - Rhino DSL 134
  - runtime AST processing 250
  - script references 96
  - self 58
  - sequences 296
  - similarity to C# syntax 302
  - singletons 25
  - slicing 296
  - standard parser 116
  - statement modifiers 304
  - statements 297
  - string interpolation 296
  - strings 296
  - structures 306–307
  - suitability for DSLs 304
  - symbols 82, 144
  - syntactic sugar 72
  - syntax 48, 285
  - syntax documenting 228
  - time spans 123
  - try ... ensure 122
  - types compared with C#
    - types 306
  - underscore character 77
  - upon keyword 274
  - using macro 120
  - variable names 287
  - website 23
  - whitespaces 303
  - why use 1
- Boo Build System 19–20
- Boo Code Builder 123
- Boo compiler 219
  - Boo.OMeta 283
  - CLS-compliant language 115
  - compiler steps 129
  - extending 33
  - extensibility 108
  - internals 103
  - pipeline 109
  - structure 109
  - See also* compiler
- Boo Lang Studio 197
- Boo on Rails 190
- Boo.OMeta 283
- booc 286
- booi 286
- booish 285, 287, 293, 298
  - indentation 289
  - multi-line comments 288
- BooJay 17
- Boolean expressions 287
- Boolean values 287
- BooleanExpression 290
- BooParser 213
  - vs executable code 216
- BooPrinter 184
- bootstrapping 63
- boundaries 99
- Brail 18, 26
  - versioning 190
  - versioning compared with
    - Binsor 191
- break statement 291–292
- breaking changes 190
- brittle language 256
- BSD license 22
- build engine 49
- build process 14
- build scripts 43
- build tool 44
- building DSLs 43, 48, 51, 63
  - styles 48
- build-script DSLs 89
- business analysts 98
- business domain 54
- business DSLs 14, 45, 48, 55, 63–64
  - building 45–46, 80
  - examples 46
  - purpose 43
- business knowledge 15
- business logic 51, 56, 78, 84, 105, 257
- business rule engines 15
- business rules 7, 46, 51, 64–65, 91
- Business-Condition DSL 258
  - creating 258
  - defines 262
  - defines compared with C/
    - C++ defines 262
  - extracting definitions 258
- business-facing DSLs 80
  - building 80
- byte 1

---

**C**

- C, text-substitution macros 112
- C# 4, 10, 18, 25, 153
  - 2.0 11
  - 3.0 11, 33
  - 4.0 24
  - compiler 23
  - pseudo C# 168
  - verify method 112
- C++ 7, 12, 98, 180
  - compiler switches 183
  - defines 262
  - text-substitution macros 112

- cache
  - invalidation 136
  - precompiled 242
- caching 60, 134–135, 146, 244
  - cache invalidation 135, 147
  - cache invalidation policy 146
  - compilation costs 135
  - Rhino DSL 97
- cascading updates 244
- case statements 303
- Castle MonoRail 190
- Castle Project 19
- Castle Windsor 45
- circuit breakers 105
  - Circuit Breaker pattern 105
- clarity 73, 284
  - issues in scaling 240
- class definition 276
- class designer 9
- class diagram 57
- class variable 299
- ClassDefinition 278
- classes 297
  - instance 298
- clear code 3, 6
- CLI. *See* Common Language Infrastructure
- client 47
- closed environment 150
- closures 13, 103
- CLR 97, 142, 283
  - assemblies 17
  - assembly language 17
  - CLR API 45
  - CLR application 55
  - CLR assembly language 160
  - CLR classes 115
  - control add-ins 99
  - naming conventions 32
  - objects 66–67
  - See also* Common Language Runtime
- Coco/R 8
- code base 3
- code completion 195, 198, 202
  - current context 207–208
  - current word 207
  - DefaultIndex property 205
  - displaying selections 206
  - generating list 208
  - hooking up events 203
- image index 208
- ImageList property 205
- location to insert selected item 205
- PreSelection property 205
- problem 203
- provider 204
- provider functionality 206
- UI 206
  - when to continue 205
- code duplication 182
- code file structure 227
- code generation 11, 216
  - DSL writers 218
- code management 111
- code string 8
- CodeCompletionWindow 203
- CodeDOM API 216–217
  - Boo implementation 216
- code-driven API 54
- code-generation DSLs 89
- CodeOriented naming 33
- collections 123
- Command pattern 73, 85
- commands 74, 227, 229
- comments 288
- Common Language Infrastructure 16
- Common Language Runtime 17, 22
  - See also* CLR
- CommonAuthorization-  
Methods.dll assembly 95
- compilation cache 137
- compilation errors 102
  - categories 104
  - handling 104
- compilation unit 123
- compile, modern compilers 128
- compiled expression 251
- compiled languages 6
- compiler 12, 252, 286
  - auto-import step 144
  - compilation process 104
  - compile-time transformation 118
  - context 129
  - creating 6
  - dealing with directly 135
- extensibility 16, 22, 24, 108
  - mechanisms 108
  - options 110
  - extensions 104
  - extensions errors 104
  - imported namespaces 121
  - inspecting runtime AST 250
  - macro statement 124
  - MacroMacro extension 118
  - macros as extensibility points 126
  - manipulation code 111
  - meta-methods 110
  - naming convention 129
  - number of times invoked 241
  - object model 117–118
  - options 44
  - parameters 95, 123, 137
  - security infrastructure 99
  - services 123
  - steps 109, 115, 128
  - structure 129
  - switch 303
  - theory 8
  - transformation 112
  - verify implementation 111
  - with block 168
  - See also* Boo compiler
- compiler pipeline 109, 123, 129, 137, 250, 261
  - compiling programmatically 132
  - compiling with custom pipeline 131
  - extension points 109
  - incorporating compiler steps 131
  - meta-methods 250
  - PrintAst step 111
- compiler steps 163, 248, 260
  - adding to compiler pipeline 129
  - clustering 145
  - code analysis 129
  - compile-time code generation 129
  - definition 129
  - examples 129
  - implicit base class compiler step 130

- compiler steps (*continued*)
    - registration 131
    - Rhino DSL 130
  - complex code 3
  - computer science 1
  - concise code 6
  - condition expression 252
  - condition statement 288
  - conditional semantics 247
  - conditionals 303
  - ConditionExpression 251
  - configuration 55
  - configuration DSL 55
  - configuration scripts 45
  - configuration tasks 45
  - connection strings 45
  - constructor method 299
  - continue statement 291–292
  - control add-ins 99
  - control blocks 26
  - control flow 50
  - control statements 288
  - convention over configuration
    - principle 88
  - convention-based methods 37
  - convention-based structure 90
  - conventions 89–90
  - copy-on-change approach 182
  - correlated macros 125
  - CPU cycles 104
  - current context 212
    - caret position 207
  - current version 182
  - custom languages 5
  - CustomizeCompiler
    - method 137
- D**
- 
- data mining
    - extracting expressions from scripts 253
  - data store 5
  - data warehouse 19
  - database
    - engine 49
    - migration 184
    - querying 5
    - storage 139
    - versioning 184
  - Davey, Andrew 20
  - Davidson, James 44
  - DDD 55, 62
    - combining with DSLs 54
    - DDD-based domain model 56
    - implementation 55
    - See also* domain-driven design
  - debuggability 283
  - debugger 142
  - debugging 73, 244, 276
    - documenting 231
  - decision-support system 19
  - declarative approach 47, 50, 78
  - declarative DSLs 48–49, 52, 61, 81, 83
    - examples 49
    - intention 48
    - operating procedure 49
    - purely declarative DSLs 50
    - when to use 49
  - DefineMacro 260
  - definition files 258, 274, 279
  - delegate 57, 69, 155, 164, 247
    - action delegate 140
    - anonymous blocks 31
    - anonymous delegate 58, 103
    - generated delegate 275
    - generating from macro statement 275
    - parameter, as 81, 163
    - when meta-method 57
  - demoware project 223
  - denial of service 99–100
  - denied-unless-allowed
    - policy 77
  - dependencies 11, 44, 80, 86
    - dependency management 44
  - dependency injection 148
  - deployment 244
    - issues in scaling 240
  - depth order 255
  - derived classes 74
  - design approach 3
  - design parameters 86
  - design pattern 73
  - designing applications 63
  - #develop 198
    - API 207
    - Boo support 198
    - caret position 206
  - code completion 203–204, 206
  - creating UIs 202
  - displaying code completion selections 206
  - ICSharpCode.TextEditor 199
  - keywords 201
  - language binding 203
  - parser 206
  - parsing rules 201
  - TextEditor 200
  - word 207
  - See also* SharpDevelop
  - developer documentation 222
  - Developer Guide
    - creating 232
  - documenting
    - advanced behaviors 235
    - behaviors
      - implementation 234
    - conventions
      - implementation 235
    - DSL 233
    - DSL implementation 232
    - external
      - implementations 236
    - external integration 236
    - external resources 232, 236
    - keywords
      - implementation 234
    - notations
      - implementation 235
    - prerequisites 232
    - syntax 232
    - syntax
      - implementation 233
    - versioning 233
  - documenting DSL
    - implementation 232
  - importance of documenting
    - notations and conventions 235
  - including code and tests 233
  - scheme 232
  - using tests as
    - documentation 232
  - development environment 14
  - development mode 244
  - diff 10

- diffing 10
- disposable code 254
- Disposable pattern 127
- distributed messaging
  - backend 83
- distribution 245
- DLL 44
- DLR. *See* Dynamic Language Runtime
- documentation 10, 221, 263, 281
  - 101 LINQ Examples 237
  - approaches 222
  - audience 222
  - comments 227
  - common operations
    - example 230
  - conventions
    - implementation 235
  - documenting
    - actions and
      - commands 230
    - AST transformations 236
    - behaviors
      - implementation 234
    - debugging 231
    - DSL implementation 232
    - external integration
      - points 236
    - prerequisites 232
    - syntax
      - implementation 233
    - tools 231
  - example scripts 224
  - examples of forgotten
    - topics 230
  - executable
    - documentation 237
  - for developers 232
  - importance of 221
  - including comments 237
  - including examples 225
  - language reference 230
  - language syntax 227
  - maintaining 238
  - making readable 224
  - multiple examples 225
  - samples 222
  - screenshots 224
  - style 226
  - syntax 227
  - syntax rules 230
  - tests as a form of 237
  - types 222
  - user guide 225
  - user guide style 226
  - using tests as 233
  - what to include in 222
- domain 225
- domain experts 42–43, 46–47, 54
- domain logic 53
- domain model 84, 156, 266, 269
  - auditable 266
- domain operations 51
- domain terms 54
- domain-driven application 54
- Domain-Driven Design 265
- domain-driven design 3, 51–53
  - ubiquitous language and DSLs 54
- domain-specific languages 1, 5
  - language types 1
- dotTrace 17
- DSL API 156
  - facades 156
  - testing 157
  - See also* APIs
- DSL engine 69, 82, 255, 267, 271, 279
  - Authorization class 158
  - modifying 175
  - nonfunctional change 175
  - optimizations 175
  - precompiled cache 243
- DSL environment 175, 177, 194
- DSL execution
  - engine 212
  - how to get results of 223
- DSL files. *See* DSL scripts
- DSL IDE
  - #develop 198
  - problems with Visual Studio 197
  - UI 213
  - using Visual Studio 196
- DSL implementation 264
  - bringing it all together 276
  - infrastructure-level
    - implementation 278
  - real-world 263, 279
- DSL infrastructure 134
  - logic 136
  - requirements 135
- DSL instances 210
- DSL integration,
  - administrating 105
- DSL model and API 175
- DSL rule engine 266
- DSL scripts 276
  - breaking 151
  - directory structure 89
  - editing in production 244
  - filesystems 88
  - performance issue 87
  - reducing complexity 256
  - testing 160
  - See also* scripts
- DSL segregation 102
  - options 101
- DSL syntax 14, 175–176, 222, 271
- DSL types 7, 10, 20
- DSL writer 217
- DslEngine 60–61, 69, 136, 140, 151
  - class 158
  - extension points 138
  - local cache 138
  - virtual methods 137
- DslExecuter 100
- DslFact 170
- DslFactory 60–61, 136, 155, 161, 278
- DslPreCompiler 242
- DslRunnerTestCommand 169
- DSLs 5, 7
  - accessing code at
    - runtime 248
  - adding transparency 248
  - administrating 105
  - administration 106
  - aggregated visualization 281
  - AST attributes 128
  - auditable if 246, 275
  - auditing support 267
  - avoiding implicit
    - concepts 269
  - background
    - compilation 243
  - base class 149
  - batch compilation 135

- DSLs (*continued*)
  - behaviors 271
  - benefits 84
  - best practices 257
  - beyond the code 280
  - Boo-based 160
  - building blocks 51
  - building DSLs 43, 50, 63
  - business keywords 258
  - capturing script
    - filename 248
  - challenges 239–240
    - editing scripts in
      - production 244
  - change management 106
  - code as data 212, 239
  - code generation 194, 216
  - combining with domain-driven design 53
  - command pattern, and 73
  - compilation time 240
  - components 175
  - constraints 43
  - conventions 222
  - creating a professional-level DSL 194
  - creating an IDE for 194–195
  - creating and using 263
  - creating positive first impression 223
  - creating single purpose dialects 211
  - custom DSL writers 217
  - dark side of 78
  - data as a core concept 281
  - data mining 253
  - debuggable 276
  - declarative DSLs 213
  - dependencies 149
  - deploying scripts in
    - production 239
  - design of 47, 53, 264–265
  - designing 264
  - designing a system with 64, 264
  - designing applications 63
  - dialect 212
  - dialect explosion 269
  - dialects 284
  - difference with fluent
    - interfaces 40
  - differences between API and
    - model 156
  - displaying DSL
    - execution 194
  - documentation 174, 221
  - documenting
    - implementation 221
  - DSL-integrated
    - application 86
  - editing strategy 78
  - engine 51, 94, 136
  - error management 104
  - error-handling strategy 103
  - errors 102
  - excellent language 269
  - execution of 59
  - expected usage 264
  - explicit concepts 270
  - extensibility 139
  - external inputs 281
  - extracting runtime
    - information 212
  - facade 33, 54
  - facilities for abstraction 258
  - factory 70
  - file-based solutions storage
    - medium,
      - recommendation 142
  - for testing 162
  - generators 306
  - graphical representation
    - 212
  - graphical representations for
    - textual DSLs 209
  - hybrid 50, 84
  - idioms 134–136, 143, 146
  - imperative DSLs 213
  - implementation 49, 264
    - challenges 239
      - class diagram 272
      - of keywords 164
  - infrastructure 52, 134
  - infrastructure concerns 64
  - instances 61, 70, 158
  - integrated DSLs 87
  - integrating into
    - applications 86
  - integration tests 151
  - keywords 81
  - language usage 263
  - layers 52
  - logic 72
  - maintenance 53
  - making expressive 269
  - making extensible 269
  - making the implicit
    - explicit 268
  - managing large number of
    - scripts 240
  - managing snippets of 80
  - memory usage 240
  - merging scripts 258
  - multi-dialect languages 256
  - multifile scripts 254
  - .NET 209
  - open environment 150
  - ordering 91, 135
  - organization 98
  - Oslo project 213
  - performance 86
    - considerations 96
    - issues 98
  - policy, behavior
    - separation 264
  - precompilation 241
  - problems 135
  - production-quality 135
  - real-world
    - implementation 263–264
  - reason to use 64
  - refactoring 270
  - reusing 94
  - reusing scripts 86
  - scaling 240, 262
    - complexity 240
    - implementations 239
    - issues 240
  - scheduling tasks 39, 41
  - scope 282
  - securing in applications 101
  - segregating from the
    - application 98–99
  - single language 256
  - single-file design 255
  - source control 106
  - startup costs 97
  - startup time 240
  - strategies for editing scripts
    - in production 244
  - structure 51, 86, 151
  - symbols 81
  - syntax 39, 83

DSLs (*continued*)

- tasks in
  - implementations 106
  - technical issues 240
- testable DSLs 150
- test-driven development 151
- testing 150
- textual DSLs 185
- tooling 194
- tooling support 222
- tracing 231
- tracing execution 195
- tracking script filename 276
- transparency 246
- turning implicit concepts to
  - explicit concepts 269
- types 39
- types of DSLs 43
- UI 194
- UI integration 281
- user-extensible
  - languages 239, 256
- version 1.0 193
- version 2.0 178, 181
- versioning 173
- visualization of 209
- why 13
- XML files 281
  - See also* domain-specific language
- duck types 297
- duck typing 13, 24, 34, 110
- DumpExpressionsToDatabase-Visitor 253
- dynamic dispatch 34
- Dynamic Language Runtime 8, 18
- dynamic languages 8, 12–13, 34–35

**E**

- early-bound semantics 138
- edge cases 78
- Emacs 47
- embedded DSLs 7, 21
- empty selection 208
- encapsulation 89
- end users 47
- endpoints 65, 70
  - multiple endpoints 256

- ERP systems 16
- error handling 78, 244, 259
- error management
  - emitting errors 189
  - lexical info 219
- errors
  - divide by zero error 103
  - logging 104
  - operation database 104
- ETL process 19
- Evans, Eric 53, 55, 265
- exact syntax 271
- Excel
  - macros 16
  - VBA 47
- exception policies 102
- executable
  - documentation 222, 237
  - tests 237
- executable language 54
- executable output 8
- execution behavior 264
- execution engine 6, 52
- execution environment 52, 222
- execution points 231
- execution semantics 222
- explicit concepts 269–270, 284
  - preferred customer 270
- explicit typing 23
- expressing intent 6
- expression trees 251
- expressions 286
- extensibility 6, 135, 149, 264
  - unauthorized extensions 269
- extensibility DSLs 62
  - building 47
  - implications 47
  - purpose 43
- extensibility mechanisms 47
- extensible DSLs 16
- extension methods 12, 33, 57
- extension points 138
- extension properties 34
- extensions 52
- external dependencies 148
- external DSLs 7–8, 48, 213, 282
  - tools 8
- extract, transform, and
  - load 14, 19, 55
- extract-transform-load (ETL)
  - tool 191

**F**

- facade layer 149
- facades 40, 52
- failing test 171
- Fibonacci series 301
  - numbers 301
- field declarations 306
- filesystem
  - organization 147
  - structure 88
- filtering 5
- fine-grained control 52
- Fit testing tool 171
- Fit fixtures 171
- flexibility 2, 17, 41, 46, 73
  - host language 282
  - importance of 284
  - using DSL 264
- fluent interface API 10
- fluent interfaces 7, 10, 12, 20, 40–41, 61
  - choosing over DSLs 42
  - implementation 11
  - scaling 42
- folder-hierarchy
  - convention 181
- for loops 14, 305
- for statement 291–292
- Fowler, Martin 5
- FreeBSD 2
- full-fledged language 6

**G**

- game logic 47
- garbage collection 98
- general-purpose languages 2, 4–6, 8
- GeneratePropertyMacro 77, 146
- generator expressions 301
- generators 300, 306
- generic object 35
- generics 11
- Getting Started Guide 223
  - examples 224, 227
  - high-level details 228
  - introduction 224
  - what to include in 223
- globals 259

GOLD Parser 8, 283  
 grammar 8  
 graphical DSLs 7, 9, 20, 48  
   representations  
   problems 209  
 great-migration strategy 185  
   *See also* versioning  
 GREP 28  
 grep 195  
 Groovy 180

## H

---

hashes 293, 295  
   literal hash 295  
 heterogeneous  
   environments 65  
 highest priority script 93  
 hogging the CPU 99  
 host language 6, 20, 282  
   host-language compiler 152  
 HTML DOM 113  
 HTTP server 27  
 HttpListener 27  
 Hughes, Brent 285  
 hybrid DSLs 49–50  
   declarative concepts 50

## I

---

ICompilerStep 129  
 ICompletionDataProvider 203  
 ICSharpCode.TextEditor 202  
 ICSharpCode.TextEditor  
   DLL 199  
 IDE 222  
   code completion 203  
   contextual code  
     completion 206  
   customizing for a specific  
     DSL 200  
   environment 195  
   integrating into your DSL  
     application 198  
     tooling 42  
   using SharpDevelop 198  
 idioms 134  
   finding repeated idioms 254  
 IDslEngineCache 136, 138  
 IDslEngineStorage 136, 138,  
   140–141

if statement 288, 304  
 ignored scripts 105  
 IIS 244  
 IL 95, 97, 102, 110, 112, 169  
   IL instructions 114  
   IL-based languages 160  
   *See also* Intermediate Lan-  
     guage  
 imperative DSLs 48, 84  
   build scripts 49  
   intention 48  
   operating procedure 49  
   when to use 49  
 imperative programming 50  
 implementation guide 221  
 implementation pattern 25  
 Implicit Base Class 68  
   approach 68, 85  
   pattern 143  
 implicit base class 67–68, 73,  
   81, 129, 137, 153  
   creating 130  
   OrderRule 247  
   structure 163  
   testing scripts 160  
   testing the DSL API 155  
 Implicit Base Class pattern 143  
 implicit blocks 69  
 implicit concepts 269  
   preferred customer 270  
   refactoring 270  
 ImplicitBaseClass 143  
 ImplicitBaseClassCompilerStep  
   138, 143, 278  
 imports 299  
 infrastructure 72  
 in-memory cache 138  
 integrated DSLs 87  
 integration testing 14  
 IntelliSense 11, 43  
 intention-revealing  
   programming 3, 6  
 interaction-based testing  
   Rhino Mocks 153  
 interactive shell 286  
 Intermediate Language 6, 17,  
   160  
   *See also* IL  
 internal DSLs 7–8, 10, 12, 20,  
   213  
   advantage 13

  code-related issues 251  
   testing DSLs 152  
 internal messages 66  
 Internet Explorer 14  
 interpreted languages 6  
 interpreter 6, 8, 286  
   executable file 286  
 invalid code 104  
 invasive execution 244  
 Inversion of Control 11  
   container 11, 64  
 IoC Castle Windsor  
   container 190  
 IoC container 148  
 IoC. *See* Inversion of Control  
 IQuackFu 67, 71, 110  
   interface 34–35, 37  
 IronPython 18  
 IronRuby 18  
 is keyword 302  
 is not keyword 302  
 isa keyword 302  
 isolation 102  
 IT DSLs 16  
 iterations 305  
 iterative design 53

## J

---

Java 3–4, 10  
 JetBrains 17  
 JIT 23, 97  
 JRuby 180  
 JSON 66–67  
   formatted messages 65  
   JSON endpoint 70  
   JSON messages 72  
   JsonMessageAdapter 70  
   object 65  
   string 65

## K

---

keyword, and 303  
 keywords 163–164, 201, 227,  
   229  
   documenting  
     implementation 234  
 Knuth, Donald 96

**L**


---

lambda declaration 41  
 lambdas 12  
 language extensibility 10  
 language hopping 43  
 language reference 227, 230  
 language semantics 48  
 language syntax 12, 225, 227  
   difference with language  
     reference 230  
   documenting 221  
 language-oriented API 54  
 language-oriented  
   programming 7, 17, 41,  
     43, 53  
   noise reduction 41  
 late-bound semantics 34–35,  
   138  
 law of unintended  
   consequences 143  
 LDAP directory 105  
 leaking memory 99  
 Lex 8  
 lexer 198  
 lexical info 103, 219  
   LexicalInfo property 120  
 limitedTo macro 123–125  
 LINQ 11, 251  
   query 12  
 LISP 47  
 Lisp 12, 16, 112  
 list comprehension 300  
 list generator 300  
 lists 293  
   literal list 293  
   slicing 297  
 logging information 105  
 logic system 79  
 loops 288, 305  
 low-level code 2

**M**


---

M language 283  
   *See also* Oslo Modeling Lan-  
   guage  
 machine code 6  
 machine level 3  
 MacroExpander 260

macros 13, 146, 163, 312  
   block 259  
   correlated macros 125  
   debug macro 120  
   define macro 260  
   level 86  
   limitedTo macro 123  
   macro block 119  
   macro class 118  
   macro statements 118, 259  
   macro transformation 118  
   MacroMacro macro 118–119  
   nested macros 124–125  
   print macro 119–120  
   unroll macro 116, 118  
   UponMacro macro 274  
   using macro 120–121  
   whenExceeded macro 124  
 MacroStatement 119, 121–122,  
   260  
 mainstream languages 12  
 maintainability 6, 96, 283  
 maintainable code 2–3  
 maintainable language 258  
 maintenance costs 3  
 Make 49  
 malicious actions 99  
 malloc method 2  
 malware 99  
 manageability 98  
 mandatory testing 171  
 manual filters 5  
 maps 9  
 market conditions 264  
 MarshalByRefObject 244  
 masked-input validation  
   facilities 5  
 MaskedTextBox 5  
 memory  
   allocation 3  
   consumption 135  
   issues 98  
   leaks 98  
   manipulation 2  
   pressure 97–98  
   usage 240  
 memory usage 240  
 message  
   dispatch 69  
   not understood 34  
   routing 256  
   translation 256  
   translation and routing 65  
 Message-Routing API 155  
 Message-Routing DSL 65, 80,  
   92, 102, 151, 153  
   base class 67  
   creating 65  
   designing 65  
   engine 69–70, 89  
   folder structure 255  
   implementation of 66–67  
   integration into  
     application 87  
   multifile scripts 254  
   multilanguage 256  
   RoutingBase 68  
   scripts directory 87  
   simplicity 72  
   structure 103  
   syntax 66  
 message-routing script 89  
 messages 89  
   message types 89  
   message versions 89  
 messaging structure 88  
 Meszaros, Gerard 159  
 meta-methods 18, 109–113,  
   132, 246  
   anonymous blocks 58  
   compared with AST  
     macros 115, 122  
   compiler pipeline 250  
   versioning 188  
   when keyword 247  
   when meta-method 247, 249  
 method chaining 11  
 method declarations 306  
 method missing 34–35  
 methods 297  
 metrics 254  
   business-quality metrics 254  
   code-quality metrics 254  
 micro level 86  
 Microsoft 8–9  
 M 283  
 Oslo Modeling  
   Language 213  
 migration wizard 184  
   issues 184  
 migrations. *See* versioning  
 model 51, 225

modules 259  
 MSBuild 13  
 multifile DSLs 255  
 MVC frameworks 190

## N

---

Name Resolution Service 123  
 named tasks 40, 47  
 namespace imports 60  
 namespaces 144, 297  
 naming collisions 167, 202  
 naming conventions 32, 88–89, 106  
   CLR 145  
   creating language  
     reference 231  
   extension methods 33  
   implicit base class 81  
   .NET naming convention 81  
   underscores 275  
 naming issues 195  
 naming styles 33  
 naming-convention  
   approach 92  
 NAnt 13, 19, 49  
 natural language processing 7  
 .NET 3, 8, 11, 23, 28, 102  
   Common Language  
     Runtime 17  
   framework 5  
   IDE 198  
   platform 17  
   Reflector 111, 131  
   space 8  
   types 293  
   versioning 188  
 non-DSL sources 284  
 not keyword 303  
 notations 227, 229  
 null checking 14  
 Null Object pattern 252  
 NullReferenceException 251

## O

---

object graph 49  
 object lifetimes 11  
 object models 83, 109, 266  
 object-oriented language 22, 297

objects 298  
 of keyword 270, 273  
 Office 16, 47  
 Oliveira, Rodrigo 22  
 OMeta. *See* Boo.OMeta  
 one-off solution 63  
 OOP principals 68  
 open language 282  
 open source projects 23  
 Operation property 74, 76  
 OperationMacro 77  
 operations 64, 73, 227  
   authorizable operations 90  
 operator overloading 11  
 operator precedence 13  
   semantics 7  
 optimized query plan 5  
 or keyword 303  
 or operator 287  
 order authorization 265  
 ordering 91, 135, 231  
   by name 92  
   external configuration 94  
   handling externally 93  
   without order 91  
 ordering conventions 106  
 order-management system 283  
 Order-Processing DSL 249, 258  
   definitions 260  
   design 267  
   explicit concept 269  
   explicit order state 268  
   implementation 271  
   initial syntax 268  
   system design 265  
   transparency 246  
   UI integration 281  
   usage 278  
 order-processing rules 46  
 order-processing system 264  
 OrderRule 247  
 OrderRuleDslEngine 261  
 Osherove, Roy 159  
 Oslo Modeling Language 213

## P

---

parentheses 30, 32, 41, 112  
 Parr, Terence 283  
 parser 8  
 partial updates 245

Pascal case 32  
 Pascal casing 275  
 pass keyword 302  
 pattern recognition 213, 215  
 PDB files 276  
 performance 96–97  
   assembly leaks 243  
   memory leak 243  
 performance issues 244  
 permissions 72, 99  
   AppDomain 101  
 persistence 65  
 PHP 18  
 pipeline 109  
   *See also* compiler pipeline  
 plug-in code 102  
 policy 64–65, 72  
   denied-unless-allowed  
     policy 77  
   policy decision 264  
 precompilation 97, 244  
 PrecompiledCache 242  
 priming 78  
 print macro 119, 124  
 print statement 287, 289  
 prioritization 93  
 problem domain 4  
 production  
   application logs 104  
   code 106  
   database 245  
   environment 42  
   mode 244  
 productivity 2  
 professional DSLs 194, 219  
   error management 219  
 professional UI  
   DSL engine 212  
   exposing the DSL model 212  
 professional-looking DSLs 209  
   graphical representation of a  
     textual DSL 209  
 profiler 97  
 programming languages 4  
 programming patterns 23  
 prompt function 287  
 property macros 163  
 pure declarative DSL 83  
 Python 12, 16–17, 26

**Q**

Quadrant 213  
 quality software 221  
 quasi-quotations 18, 113–114  
   complex expressions 115  
   conditional code 114  
   extending the compiler 132  
   generating blocks 116  
   incompatibility with  
     macros 126  
   limitations 124, 126  
   meta-methods, and 108  
   quasi-quotation blocks 114  
   using AST 111  
 query syntax 12  
 querying 5, 11  
 quote generation 65  
 Quote-Generation DSL 63–64,  
   78, 148, 151, 195  
   adding explanations 187  
   as XML 83  
   code completion 203–204  
   contextual code  
     completion 206  
   creating test DSL 162  
   declarative DSL 83–84  
   design of 78  
   Developer Guide 234–235  
   documentation 226  
   documenting 222, 228  
   debugging 231  
   operations 229  
   file structure 228  
   implicit base class 186  
   keywords 229  
   language reference 231  
   notations 229  
   script structure 228  
   syntax highlighting 195, 200  
   testing 160, 163  
   User Guide 225  
   versioning 185–187  
   visualization of 209  
 quote-generation engine 82  
 quote-generation rules  
   derived dialect 211  
   evaluating 209  
   result of executing 210  
 QuoteGenerationCode-  
   CompletionProvider 204

QuoteGenerationDslEngine  
   211  
 QuoteGeneratorForUI 211  
 QuoteGeneratorRule 81–82,  
   211

**R**

Rake 12, 14, 49  
 range 294  
   method 294  
   numeric ranges 294  
 readability 11, 56, 82  
 readable code 2–3, 6, 10, 30  
 real-world DSL 265, 284  
   implementing 268  
   order authorization 265  
   shopping cart  
     modifications 265  
   testing 280  
   tests 280  
   tools integration 280  
   UI 280  
 real-world system  
   DSLs 264  
   implementing behavior 264  
   policy 264  
   system behavior 267  
 recurrence pattern 47  
 refactoring 262  
 reference equality operator 30  
 reference expressions 260  
 ReferenceAggregatorVisitor  
   252  
 Reflector 112, 143  
 regressions 151, 187  
   regression bugs 150, 193  
   regression test suite 178  
   test suite 174, 177  
   tests 151  
 regular expression approach 5  
 regular expressions 4–5, 7, 12,  
   49  
 relational database 19  
 repetitive code 25  
 ReplaceDefinitionsWith-  
   Expression 261  
 repository 245  
 response time 240  
 reusing code 94  
 reversibility 245

RevisitSerializedExpression  
   250  
 Rhino DSL 59, 67, 76, 82, 92,  
   97  
   AutoImport 144  
   background  
     compilation 243  
   batch compilation 88, 146  
   batching 97  
   bug fixes 135  
   building compiler step 130  
   building DSLs 132  
   caching 146, 241, 243  
   caching, using precompiled  
     assembly 242  
   common DSL idioms 143  
   compilation costs,  
     reducing 135  
   error management 219  
   external script  
     dependencies 148  
   idioms 146  
   IDslEngineStorage 139  
   infrastructure 147, 149  
   instance management 148  
   overriding cache  
     behavior 242  
   reasons for use 135  
   reducing compilation costs,  
     reducing 146  
   reusable idioms 143  
   storage 139  
   structure 134, 136  
   testability 152  
   UseSymbolsStep 144  
   warnings 220  
 Rhino ETL 14, 19  
   syntax changes 191  
   versioning 191  
 Rhino Mocks 153  
   testing DSL syntax 155  
 Router class 67, 69, 87  
 Routing DSL  
   design of 65  
   documenting 227  
   testing 152  
 routing modules 65–66  
 routing rules 255  
 routing scripts 89, 255  
 RoutingBase 70, 154  
 RoutingDslEngine 76, 154

rSpec 7  
 Ruby 12, 16, 26, 37  
 Ruby library 7  
 Ruby on Rails 7  
 Ruby-based DSL 13–14  
 rule engine 267  
   syntax 267  
 rules 13, 50, 195  
 rules engines 46  
 runtime engine 51  
 runtime errors 102–103  
   handling 102

## S

Sarbanes-Oxley tracking  
   system 54  
 scaling DSLs 240  
   clarity issues 240  
   deployment issues 240  
   technical issues 240  
   transparency issues 240  
 Scheduling DSL 39, 42, 47–48,  
   57, 68, 113  
   documenting behaviors  
     implementation 234  
   implementing 56  
   meta-method 58  
   running 59, 61  
   translation to C# 58  
 scheduling engine 61  
 scheduling implementation 40  
 scheduling semantics 40  
 scheduling system 39  
 SchedulingDslEngine 60  
 Scheme macros 112  
 script prioritization  
   disadvantages 93  
   duplicate priorities 93  
   priority order 93  
 scripting 63  
 scripts  
   administrating 105  
   background  
     compilation 243  
   batch compilation 142  
   caching 243  
   capturing filename 248  
   change notifications 138  
   compiled assembly 95  
   compiling 135  
   compiling in batches 241  
   configuring execution  
     order 106  
   data mining 253  
   debugging 142  
   dependencies 91, 94  
   diffing changes 142  
   faulty DSL scripts 104  
   filesystem 147  
   highest priority script 93  
   how to deploy 222  
   how to execute 223  
   ignored scripts 105  
   malicious script 99  
   names 142  
   numbering system 92  
   order of execution 91  
   ordered list 94  
   precompilation 241  
   prioritization 92  
   prioritization  
     disadvantages 93  
   reuse 94  
   script  
     changes 102  
     compilation 96–97  
     discovery 135  
     execution 96–97, 147  
     management 97–98  
     priorities 91  
     references 95–96  
     versions 181  
   slow DSL scripts 97  
   source control 142  
   state changes 92  
   storage 139  
   storing in filesystem 142  
   suspended scripts 105  
   top-ranking script 93  
   updating 106  
   XML file 139, 149  
   *See also* DSL scripts  
 security 72  
   code access security  
     policies 100  
   infrastructure 73, 99  
   measures 98  
   rules 91  
   systems 72, 78  
 selection criteria 279  
 semantic model 8  
 semantics 248  
   complex 248  
 semicolon 28  
 separation of concerns  
   principle 255, 258  
 serialization format 209  
 serialized AST nodes 250  
 service level agreement 123  
 Sethi, Ravi 8  
 SharpDevelop 111, 198  
   code completion 203  
   embedding in your  
     application 199  
   extending to understand a  
     DSL 200  
   *See also* #develop  
 Shopping Cart DSL 267–268,  
   278  
   AbstractCustomerPolicy 272  
   AddFileNameProperty 272  
   BusinessConditionDslEngine  
     272  
   CustomerPolicies 272  
   CustomerPolicyDslEngine  
     272  
   example 271  
   explicit concept 269  
   test for language feature 280  
   test suite 280  
   Treatment 272  
   TreatmentOfToMethodCall  
     272  
   UI 281  
   WhenMacro 272  
 shopping cart  
   modifications 265  
 significant-whitespace mode 26  
 simple code 3  
 simplicity 2  
 single responsibility  
   principle 255  
 single-purpose language 7  
 single-shot strategy 179  
   *See also* versioning  
 Singleton pattern 24, 127  
 SLA  
   macro building 123  
   violations 123  
   *See also* service level agree-  
     ment

- slicing 296
- Smalltalk 12, 16
- smart factories 11
- software design 53
- software problem 5
- source control 10, 142, 195, 245
  - system 181
  - use in production 245
- source database 19
- source files 276
- specialized tasks 5
- specialized tools, challenges 5
- specialized vocabularies 5
- Specter 20
- SQL 7, 9, 43, 49
  - querying code structure 254
- SQL Server Integration Services 10
- stack 20
  - overflows 100
- stagnant DSL 257
- startup performance 241
- startup time 240–241
- statement block 289
- statement modifiers 31, 305
- StatementBlock 290
- static gateways 148
- static strings 30
- static variable 299
- statically typed language 22
- string builders 29
- string interpolation 29
- StringBuilder 30
- strings 296
- strongly typed wrapper 35
- structured message format 89
- StructureMap 11, 148
- stubbed class 154
- StubbedRoutingBase 154–155
- StubbedRoutingDslEngine 154
- suspended scripts 105
- switch/case statement 304
- symbols 81
  - compiler step 144
- syntactic flexibility 10, 12
- syntactic noise 30
- syntactic sugar 29, 48, 57
- syntax 4, 8
  - errors 104
  - extensions 51

- implementation 233
- rigidity 12
- syntax highlighting 195, 198, 200–201
  - complexity 196
  - customizing 200
  - hand rolled 195
  - SharpDevelop 198
  - strategy 202
  - Visual Studio 196
- syntax rules 227
  - documenting 230
- system configuration 43
- system transparency 246
- System.AddIns 102
- System.CodeDOM 113
- System.Console.WriteLine 299
- System.DateTime 76
- System.Windows.Forms
  - namespace 25

## T

---

- targets 44
- tasks 2, 13, 19
  - repeatability 40
- TDD. *See* test-driven development
- team environment 10
- technical domain 43
- technical DSLs 14, 48, 55, 64, 66–67
  - advantages 45
  - building 43, 45
  - common use 63
  - disadvantages 45
  - purpose 43
  - quote generation 80
- technical issues in scaling 240
- templating 55
  - DSL 48, 55
- test case 160
- test cycle 42
- testability 150–151
- testable DSLs 150
- testable systems 150
  - maintainability costs 150
- test-driven development 151, 178
- testing 284
  - application-testing DSL 171
- assertions 161
- interaction-based 153
- mandatory testing 171
- primary DSL 160
- secondary testing DSL 160
- test case 155
- testing the API 155
  - with C# code 162
- Testing DSL 150, 157, 162
  - approach in building 162
  - executing 165
  - extending 165
  - initial syntax 162
  - integrating with unit-testing framework 166
  - precompile step 166
  - primary DSL 162
  - property macros 163
  - recursive testing DSLs 162
  - testing a script with logic 165
  - testing DSL to test 162
  - verification keywords 164
- tests 151
  - abstractions in 162
  - acceptance tests 171
  - basic 151
  - breaking test 178
  - creating test DSL 162, 171
  - dependence of language
    - tests on policy decisions 280
  - dummy test 171
  - effects on versioning 174
  - engine 178
  - failing test 171
  - for API 155
  - for Engine 158
  - for language
    - implementation 280
  - for policy decisions 280
  - for scripts 160
  - for syntax 152
  - for the API 178
  - for the syntax 178
  - importance of 280
  - integrating with XUnit.NET 166
  - integration tests 178
  - interaction based 153
  - language tests 280

tests (*continued*)

- mandatory tests 171
- multiple tests 167
- no-cost approach to
  - building 160
- quality of software 172
- reasons to have for DSLs 150
- regression 151, 174, 178
- syntax in external DSL 152
- test code 162
- test methods 169
- to verify conventions 178

text editor 42

text generation 49

text processing 4–5

- specialized tool 4

thread exceptions 100

thread-safe 243

time changing system 3

TimeSpan object 123

tool chain 43

Tower of Babel strategy 181

- code duplication 182
- copy-on-change
  - alternative 182
- drawback 181
- variation 182

*See also* versioning

trace viewer 222

tracing 267

transaction-safe manner 5

translators 67

transparency 246, 248

- issues in scaling 240

treatment of keyword 278

treatment of statement 273

triggers 50

true audit log 276

try ... ensure block 127

try ... except statement 125

tuning 78

Turing, Alan 99

type declarations 306

type definitions 259

type inference 23

Type System Service 123

types 293

- declaring 297

## U

---

ubiquitous language 54–55

UIs 78, 106, 209, 223, 245, 251

- approaches to creating 220
- BooParser 213
- DSL code to 215
- feeding the DSL writer 218
- functional design 209
- good UIs 64
- integration 284
- issues 10
- mockup 281
- pattern recognition 216
- quote-generation UI 79
- to DSL code 216
- UI dialect 211

Ullman, Jeffrey 8

UML 9

- UML diagram 9

underscores 145

unexpected token 104

unit testing 49, 159, 161

- extensibility mechanism in
  - frameworks 166
- unit-testing framework 166, 169

unless keyword 304

upon keyword 274

upon macro 278

UponState method 274–275

usable DSL 272

user documentation 222

User Guide 225

- actions 229
- actual details 228
- Boo syntax 228
- commands 229
- common operations 230
- documenting notations 236
- documenting syntax 233
- domain 225
- explaining dealing with the
  - unexpected 231
- keywords 229
- language syntax 227
- notations 229
- Quote-Generation DSL 226
- role 225
- syntax rules 230

troubleshooting

- information 232
- what to include 225

user-defined abstractions 257

user-editable script 257

user-extensible languages 256

user-level debugging 225

UseSymbolsStep 145

using macro 121–122, 124

using statement 312

## V

---

VB.NET 18, 26, 28, 55

- VB.NET 9 12

VBA 16, 47

verify method 113

versioning 65, 173, 263

- abandon-ship strategy 179
- adapter strategy 182
- adapters 182–183
- adapting during
  - compilation 183
- additive change 180, 185
- application integration 176
- approaches 173
- automatic migrations 184
- backward compatibility 180
- baseline 174
- Binsor 190
- boundary 192
- Brail 190
- business implications 190
- considerations 173
- current version 174
- definition 175
- documentation 174
- DSL API 176
- DSL model 176
- error handling 190
- executable form 174
- for external DSLs 176
- freezing a version 182
- functional changes 175
- great-migration strategy 184
- handling breaking
  - changes 187
- in the real world 173
- incompatibility
  - strategies 180
- internal DSLs 176

versioning (*continued*)  
 logging 190  
 migration wizard 184  
 modifying DSL API/  
 model 176  
 modifying DSL engine 175  
 modifying DSL  
 environment 177  
 modifying DSL syntax 177  
 non-radical changes 182  
 planned  
 incompatibilities 175  
 Quote-Generation DSL 175  
 radical changes 182  
 real-world examples 190  
 reverse engineering 174  
 Rhino ETL 191  
 runtime adaptation to a  
 version 183  
 script versions 181  
 second version 173  
 single-shot strategy 179  
 strategies 173, 175, 179  
 strategies, application of 185  
 testing, for backward  
 compatability 151  
 Tower of Babel strategy 181  
 users experience 177  
 version 1.0 183  
 version 2.0 183  
 version 3.0 185  
 version numberings 180  
 versioning story 175  
 when to apply 192  
 versioning boundary, types 192  
 version-numbering system 180

Visitor pattern 131  
 Visual Studio 17, 203, 219  
 API 197  
 Boo support 197  
 class diagrams 209  
 developing extensions 197  
 macro feature 47  
 Visual Studio 2008 196  
 Visual Studio 2008 Shell 196  
 Visual Studio 2010 197  
 Visual Studio DSL Tools 9  
 visualization approach 9  
 Vivier, Cedric 20

## W

---

Watir 14  
 web farm 244–245  
 when keyword 275, 303  
 when macro 278  
 whenExceeded macro 124–125  
 implementation 125  
 while loops 305  
 while statement 290  
 whitespace-agnostic modes 26,  
 303  
 Wilde, Tim 41  
 Windows 285  
 Windows Explorer 23  
 Windows Process  
 Activation 244  
 Windows Workflow  
 Foundation 10  
 Windsor 148  
 Windsor IoC container 14, 282  
 WinForms 5, 25

word 207  
 current word 207  
 previous word 207

## X

---

XML 13, 44, 66, 83, 113, 281  
 #develop 198  
 #develop TextEditor 200  
 DSL code to UI 215  
 DSLs 84  
 external configuration for  
 ordering 94  
 files 10  
 one-to-one mapping 83  
 persistence format 10  
 pure declarative DSL 83  
 Quote-Generation DSL 83  
 storage file 139  
 XML noise 45  
 XML objects 72  
 XML-based storage 142  
 XmlFileDslEngineStorage  
 class 139  
 XmlMessageAdapter 72  
 XmlObject 67  
 XmlObject 24, 35–37, 72  
 xUnit  
 extending 167  
 integrating tests into 169  
 sample test 167  
 xUnit.NET 167

## Y

---

Yacc 8