

index

Symbols

`#{} (string interpolation operator)` 106
`% (modulo method)` 236
`&` prefix to Proc/lambda to use as code block 356
`*` (multiplication method) 6, 236
`+` (addition method) 6, 236
`-` (subtraction method) 6, 236
`.` (dot operator) 6–7, 100
`/` (division method) 6, 236
`<` (comparison method/operator) 237
`<% %>` (ERb tag delimiters) 30
`<%= %>` (ERb interpolative tag delimiters) 30
`<<` (append method) 236
`<=` (comparison method/operator) 237
`==` (comparison method/operator) 237
`===` (case equality operator) 237
`>` (comparison method/operator) 237
`>=` (comparison method/operator) 237
`[]=` (index setter method) 236

A

ActionController 39–40
See also controllers
ActionPack 39–40

actions, execution sequence 62
ActionView 39–40
 form helper methods 74, 136
 helper files 80–82
 helper methods 35, 136
ActiveRecord 39
 (all) vs. `find_all` 291
 automatic creation of instance methods by 338
 collections as 397
 conditions argument to `find` 291
 create (class method) 449
 delete_all (class method) 419
 find_by_field methods 291, 444
 first argument to find method 291
 illustrating receiver-changing permutations with 241–242
 library source code 457
 manual 241
 methods derived from database field names 135
 models inherit from Base 364
 objects as Ruby instances 122, 372
 Ruby lessons from collection handling 289–292
 setter(=terminated) methods in 133–136
 source code files 460
 supplies full toolkit for database record 419
 update_attribute method 418
 validation facilities 444
ActiveRecord collections
 :all modifier of find method 399, 417
 :conditions argument of find method 399
ActiveSupport library
 automatic translation of action name to filenames by 460
 dependencies.rb 459
 named-based inference 460
 provides facilities used across several other subsystems 460
Apache Web server 58, 60–61, 74
 configuration file examples 61
application domains
 actions and views closely linked 36
 choosing publicly available views of 36
 specifying actions within 35
application, connecting to 58–59
arithmetic methods (operators), summary 6
arithmetic operations 271–272
arrays 279–292
 + method 283
 == used to determine duplicates for `uniq` method 286
 []= method 280
 as anchor for more complex container classes 289

- arrays (*continued*)
 - as ordered collections 278–279
 - as resulting form of many collection operations 278
 - combining with other arrays 283–285
 - compared to hashes 278–279
 - concat method 283
 - concat vs. + 283
 - concat vs. push 283
 - conversion to string with `to_s` 242
 - creating with `Array.new` 279
 - difference between push and 282
 - each method 224, 286
 - each vs. map 245
 - each_with_index method 287
 - filtering methods 287
 - find and find_all vs. Boolean query methods 288
 - find method 287
 - find_all method 288
 - flatten method 285
 - flattening nested array of 395, 400
 - getting unique elements 395, 398, 400
 - grabbing all but last element of 406
 - how to add or remove an object 282
 - inject method 417
 - inserting items into 280–283
 - iterating through with each 244
 - joining elements with commas or comma-space 286
 - joining into string 406
 - manipulating beginning and end of 282
 - mapping operations 245, 400
 - most basic Ruby container methods 289
 - negative indices count from right 406
 - nesting 279, 285
 - of countries as part of musical periods 411
 - pop method 282–283
 - push method 282
 - query methods except size return Boolean 288
 - replace method 284
 - resulting from searching ActiveRecord collections 400
 - return values of filtering methods 287
 - reverse method 285
 - reverse! (in-place reverse) method 286
 - second argument to `Array.new` repeated 280
 - select method (synonym for find_all) 288
 - setting or getting multiple elements 281
 - shift method 283
 - shuffling 365
 - subclassing 289
 - supplying a code block to `Array.new` 280
 - transforming 285
 - uniq method 286
 - uniq! (in-place uniq) method 286
 - unshift method 282
 - versatility 158
 - zero-indexed 281
 - zip method 284
 - ASCII value, generated by `String#each_byte` 307
 - associations (ActiveRecord) 48
 - associations.rb (Rails source file) 457
 - belongs_to 48
 - belongs_to as sample source code 457
 - configuration-like syntax 73
 - has_many 48
 - in source-code tree 457
 - inter-entity modeling facility 48
 - source code files for 461
 - `attr_*` methods 136
 - as self-documenting 138
 - `attr_accessor` 138, 148
 - `attr_reader` 138–139, 157
 - `attr_writer` 138–139
 - automatic creation of methods with 138
 - attributes 136–140
 - commonness of 137
 - set of techniques for creating 136–139
- ## B
-
- bang (!) methods 240–241
 - among string methods 260
 - dangerous 240
 - in built-in classes 240
 - not the only methods that change a receiver 260
 - receiver-changing 240
 - barewords
 - as arguments to require 22
 - how Ruby interprets significance 119
 - method calls 7
 - black box 4, 48
 - Boolean objects 234, 245–250
 - as special objects 248
 - false 106
 - in relation to Boolean values 249
 - vs. Boolean states 245
 - Boolean values 106–108, 246
 - every object has one 245
 - examples of mapping of expressions to 249
 - in relation to Boolean objects 249
 - vs. Boolean objects 245
 - breadth-first 4, 68
 - Builder 53
 - built-in classes 234
 - can be opened as easily as new classes 365
 - changing or amending 366
 - circularity of hierarchy 145
 - literal constructors 235
 - most instantiated with new 234
 - `NilClass` 249
 - not instantiable with new 235
 - syntactic sugar method naming in 237

- built-in methods
 - String#to_i 6, 13
 - built-in modules 155
 - See also Comparable, Enumerable
 - built-in top-level methods 204
- C**
-
- C extensions 22
 - written for speed 23
 - C++ 97
 - callable objects 351–359
 - includes Proc objects, lambdas, and methods 351
 - callbacks 359–365
 - ActiveRecord and Rails technique 365
 - append_features (included) 363
 - automatic extension 360
 - before_create 78, 83, 89
 - cascading of
 - Class#inherited 363
 - Class#inherited 363–365
 - examples of events 359
 - included (class method of Module) 462
 - method_missing 360–361
 - Module#const_missing 365
 - Module#included 361–363
 - triggered by events 359
 - capabilities of an ActiveRecord model instance
 - added programmatically 373
 - automatically created based on database structure 373
 - capabilities 372
 - derived from inheritance 373
 - inherited 374
 - semi-automatically created by association directives 373
 - capturing submatches with parentheses 319–320
 - cargo hold, as example of stack 161
 - case equality
 - === operator 212
 - defining programmatically 213
 - different for different classes of objects 213
 - case statements 80, 211
 - translated into === calls 212
 - Celsius-to-Fahrenheit conversion formula 101
 - century, calculating from year 409
 - CGI library 22
 - CGI variables, id 52
 - Class
 - as instance of itself 145
 - subclass of Module 155
 - See also classes
 - class definitions 122–130, 247
 - broken up across multiple files 125
 - empty 247
 - reopening 124
 - reopening, in Rails source code 174
 - class methods 140–145
 - as singleton methods added to class objects 143
 - at more abstract level than instance methods 419
 - conformity of with regular method-call syntax 140
 - criteria for writing 419
 - File.dirname 461
 - for searching all existing records of a class 419
 - in object-oriented languages 143
 - inherited (class method of Module) 464
 - origins as module instance methods in Rails source code 462
 - soft model enhancement with 419–421
 - vs. instance methods 143–144
 - when and why to write 141–142
 - class variables
 - not actually class-scoped 364
 - recognizable by @@ prefix 364
 - scoping rules may change in future versions of Ruby 364
 - use of in ActiveRecord source code 364
 - visible in class and method definition bodies 364
- classes
- ancestors 149
 - as method 122
 - as object factories 127
 - as objects 122, 140–141, 143–144
 - as specialized modules 155, 175
 - instantiating 158
 - nouns for names 159
 - querying for instance and/or class methods 253
 - singleton 338
- closures
- analogous to packed suitcase 353
 - defined 353
 - Proc objects as 352
- code blocks 215
 - argument syntax differs from that of method calls 221
 - converting into Proc object in method body 356
 - curly braces as delimiters 215
 - delimiters 219
 - do/end delimiters 215
 - in loops 215
 - partnership with methods 223
 - passing arguments to 220
 - returning a value from 221
 - returning values to methods that yield to them 222
 - using different ones with same method 223
 - why use? 219
- collections and containers
- arrays and hashes as main ones in Ruby 278
 - objects in their own right 278
 - searching 278
 - sorting 278, 307–311
 - ubiquity of in Rails and elsewhere 278
- command-line switches and flags 16–19
- c (check syntax) 9, 16
 - combining 19

- command-line switches and flags
 - (*continued*)
 - common 16
 - cw (check syntax in warning mode) 19, 225
 - e (execute) 17
 - l (line mode) 17
 - r (require) 18
 - v (verbose) 18
 - ve (verbose/execute script) 19
 - version 19
 - w (warn) 9, 17
- Comparable module 251, 303
- comparison of objects 234, 251–253
- == 6
- == method frequently redefined 252
- Comparable module and 252–253
- determining identity of two objects with equal? 251
- equality tests of instances of Object 251
- family of > <= methods 252
- objects born created with comparison abilities 251
- piggybacking on another class's method 253
- pre-defined return values of method 253
- redefinition of == and eql? methods in descendants of Object 251
- spaceship method (<=>) as basis of Comparable module methods 252
- conditionals 84, 207–214
 - case statement 211–214, 406
 - conditional modifiers 211
 - else and elsif 208
 - empty string 408
 - fundamental to programming 207
 - hypothetical examples of 207
 - if 6, 107, 111, 208–211
 - if clause on single line 208
 - in ERb templates 442
 - parallel to real-life decision-making 207
 - unless 209
 - unless/else sometimes awkward 210
 - usefulness of Boolean objects for 106
- configuration files 74
- constants 123, 145–148
 - actually changeable 123
 - as class names 123
 - built-in 146
 - class/module ambiguity in notation 175
 - pre-defined 146
 - reassigning vs. modifying 146–148
 - rules for naming 145–146
 - used for data storage 145
 - visibility 145
- constructors 122, 293
 - as distinguishing feature of class objects 141
 - creating classes with 144
 - initialize 128
- context, importance to meaning of Ruby expressions 178
- control flow techniques
 - case statement 211
 - conditional execution 207
 - conditional modifiers 211
 - exceptions 207
 - iteration 207
 - iterators 219
 - looping 207
 - loops 215–218
 - while/until 215
- controller actions 62–65
- controllers (ActionController)
 - adding functionality to 79–80
 - application_controller.rb generic controller 441
 - as Ruby objects 150
 - automatic creation of specified action methods 51
 - creation of 150
 - error reporting 443
 - inherit from ActionController via ApplicationController 426
 - inheritance hierarchy 441
 - logic of sorting in 79
- converting legacy data from YAML 87
- core functionality, adding to 365
- create_table 248
- customer
 - adding items to cart 449
 - adding orders to cart 446
 - breaking of tie between favorites 451
 - check-out process 418
 - completing orders 446, 449
 - controller 446
 - determining composer and instrument rankings 414
 - ID stored in instance variable 445
 - logging out 445
 - order history 414, 451
 - personalizing page for 450–453
 - ranking of favorites 450
 - showing 414, 450
 - tracking number of copies ordered per edition 416
 - tracking via instance variable 442
 - view_cart action 446
 - viewing shopping cart 446
 - See also* customer model
- customer model 438–445
 - check_out method 450
 - combined rank method for instruments and composers 415
 - example of instrument history 415
 - granularity of ranking and favorites algorithm 416
 - handling of a tie in rankings order 415
 - security issues raised by 438
 - trying out rankings methods in irb 416
 - walk-through of favorites scenario 415
 - See also* customer
- customization, usefulness of for models 437

D

database design 35
 database.yml (Rails application configuration file) 45, 75–76, 87
 dates, UNIX-style output of 125
 DateTime class 272
 debugger 27
 breakpoints 27
 stepping through lines 27
 vs. inserting debug instructions 28
 DeckOfCards class (example) 289
 delegation techniques, as alternatives to
 method_missing 361
 delete (ActiveRecord::Base method) 375
 Dir (Ruby class) 87
 dispatchers 62
 domain modeling 35
 diagraming a domain 43
 range of actions possible in domain 35
 simple approach to diagraming 43
 translating into SQL 46
 domain-specific languages 69
 explained 70
 in relation to Ruby syntax 71
 poker sample in Ruby 70
 Rails as one 71
 relative ease of learning for non-programmers 71
 Ruby as host language for 70
 specificity manifested in terminology 72
 written in existing languages 70

E

editions (of musical works)
 naming after title of collection of works 409
 prettified title 409
 tracking number of copies ordered 416
 Enumerable module 278, 303–307

all? method 304
 any? method 304
 different each methods
 behave differently 304
 each method written separately for each class 304–306
 each_with_index method 287
 find method 287, 304
 find_all method 288
 grep method 334–335
 map method 304
 methods based on underlying each method 303
 mixed into Array and Hash classes 303
 reject method 288, 304
 select method (synonym for find_all) 288, 304
 equal sign (=) in method names 131–133
 equality tests 251
 ERb (Embedded Ruby) 29, 39, 51, 53, 200
 central to Rails framework 31
 demonstration 30
 erb (command-line utility) 30
 preparation of files in 30
 Ruby code in HTML vs. HTML in Ruby code 29
 tag delimiters 30, 426
 testing hash-style arguments with 302
 erb (ERb command-line utility) 30
 errors and exceptions 143, 225–230
 adding string to integer 243
 ArgumentError 112, 226
 common exceptions 226
 creating your own exception classes 207, 228–230
 descending classes of Exception 225
 descriptive names of used by Rails 229
 exception as instances of Exception and descendants 225–227
 exceptions are objects 207
 in the Rails framework code 229

intercepting with rescue
 keyword 225–226
 IOError 226
 NameError 120, 226
 NoMethodError 110, 226, 366
 pinpointing rescue operations through exception names 229
 raising an exception, possible outcomes 225
 raising exceptions
 explicitly 227–228
 re-raising exceptions 228
 rescue blocks 226
 rescuing specific exceptions 227
 RuntimeError 226
 self-documented code through exception names 229
 TypeError 226
 ZeroDivisionError (example of triggering) 225
 eval family of methods 337–351
 access to variables in surrounding scope in
 class_eval block 350
 class_eval 349–351, 462
 class_eval vs. class keyword 350
 evaluating strings with
 class_eval 350
 instance_eval 349
 opening definition of anonymous class with
 class_eval 350
 extension vs. library 22
 extensions 21
 C-language 22–23
 writing and sharing 23

F

Fahrenheit-to-Celsius conversion formula 142
 false
 Boolean value 249
 false as keyword 250
 false as object 248
 FIFO (first in, first out) 157
 file I/O 141

file modes, w (write) 13
 filehandles 142
 filters (ActionController)
 before_filter 441, 443–444
 except modifier 443
 only modifier 444
 security and 441
 usefulness even for relatively
 safe actions 441
 floating-point numbers 104
 complexity and
 quirkiness 104
 printing to n places 428
 forms, populating ActiveRecord
 objects from fields of 135
 frameworks
 application programmer’s
 role in using 34
 computer application vs.
 house 35
 derivation of term 34
 overview 34–38
 vs. scaffolding 34

G

generate script 175–176
 global variables, (runtime
 library load path) 461
 grep 174
 finding methods in Rails
 source code files with 458
 Ruby substitute for 174

H

hashes 292–307
 => separator for key/value
 pairs 293
 adding every new key to hash
 with code block 296
 allow key-based lookup
 operations 292
 as arrays with arbitrary
 indices 278
 as unordered collections 278
 CGI library uses 279
 clear method 298
 code block, supplying to a new
 constructor 295
 combining 296–297

creating 293
 creating with []
 constructor 293
 deciding what should be keys
 and what values 412
 default code block
 technique 296
 default method 295
 default value nil, by
 default 295
 duplicate values when
 inverting 297
 each method 298
 each_key method 299
 each_value method 299
 empty? method 300
 fetch method 295
 filtering 298
 find method 299
 four names for key-testing
 method 300
 get and set operations
 294–296
 has_key? method 300
 has_value? method 300
 Hash.new constructor 293
 in method argument
 lists 301–303
 in Rails method calls 301
 include? (synonym for
 has_key?) 300
 invert method 297
 iterating over 298–301
 key/value pairs 292
 key/value structure 278
 key? (synonym for
 has_key?) 300
 keys overwritten if added
 twice 294
 length (synonym for size) 301
 literal constructor convenient
 for hashes that won’t
 change 293
 member? (synonym for
 has_key?) 300
 merge (non-nondestructive
 combination method) 297
 merge! (synonym for
 update) 297
 non-existent keys not automat-
 ically added when used 295

of state (i.e., USA) names and
 abbreviations 292
 priority of keys in merge
 operation 297
 querying 298–301
 Rails methods use hashes as
 arguments 279
 replace method 298
 return array from sort
 operations 307
 self and key yielded to default
 code block 296
 size method 300
 transforming 297–298
 two-element (key,value) array
 yielded by iterator
 methods 298–299
 uniqueness of keys 294
 update (destructive combina-
 tion method) 296
 use in Rails framework 279
 value? (synonym for
 has_value?) 301
 values don’t have to be
 unique 294
 values_at method 295
 vs. arrays 294
 helper (ActionController class
 method)
 and visibility of helper files to
 other templates 426
 helper files
 as extension of helper
 methods Rails provides 82
 created automatically 81
 example from RCRchive 81
 save code repetition 81
 vs. putting code in view
 template files 82
 helper files (ActionPack) 80–82
 contents of newly-created 425
 generic file
 application_helper.rb
 426, 428
 instance methods in callable
 from view templates 425
 naming conventions 425–426
 using from another
 directory 426
 helper methods
 custom, organizing and
 accessing 425–427

helper methods (*continued*)
 defining for view
 templates 424

helper methods (ActionPack)
 callable from view
 templates 425

end_form_tag 439

form_tag 439

password_field 439

stashing in
 application_helper.rb vs.
 distributing across helper
 files 426

text_field 136, 439

I

if
 as tool for testing Boolean
 value of any expression 247

bread-and-butter tool of con-
 ditional execution 208–211

careful placement of
 ‘end’ 210

else and elsif 208

elsif branches, clauses 209

if/else ambiguity 210

in C vs. Ruby 210

tips on using ‘else’ 209

include (Module class
 method) 159

include, mixing modules into
 classes 156

[] (index method) 236

index.html, deleting Rails
 default 58

inheritance 121–153
 as key principle in Rails
 design 149

as reflection of general-to-spe-
 cific relations 148, 151

cascading of instance method
 access through 148, 150

in object-oriented
 programming 148

inject method, for incremental
 accumulation of results 417

input
 file 11–14

gets 6, 11

keyboard 11–14

instance methods 123, 156

Array#unshift 461

notation for referring to 143

vs. class methods 143

instance variables 126–130
 and object state 126, 130

matching names with method
 names 129

names start with @ 127

persistence across method
 calls 128

visibility 127

instruments, musical, algorithm
 and specifying order 405

integers 178

chr method 307

no automatic conversion to
 string 243

introspection 111

methods (method) 108

irb (Interactive Ruby) 5, 16, 20
 as alternative Ruby
 interpreter 5

breaking out of with Ctrl-c 21

calculator-like behavior 20

exiting with Ctrl-d 21

learning Ruby with 5, 20

printing strings vs. echoing
 string expression
 values 259

prints value of each
 expression 107

Rails application console 85,
 89–90, 290

starting a session 20

testing Ruby code with 20

verbose output from 290

iterators 219–224, 244–245

basics of yielding to a
 block 219–222

basis of for keyword in itera-
 tion with each 223–224

code block execution vs.
 return value of 245

code blocks and 219

commonplace in built-in
 classes and modules 244

defined as methods that yield
 values to code blocks 222

different from non-iterators
 but in an additive way 244

each vs. for 224

each vs. map 245

figure prominently in
 Ruby 207

looping with 215

moving control from one
 scope to another with 219

multiple iterations 222–223

provide a return value 244

relation to method calls 219

yield keyword 219

yielding arguments to
 block 221

K

Kernel#inspect 331

keywords 119, 208–209

alias 366

barewords interpreted as 119

break 216

case 212

class 123, 155, 178

def 99, 123, 178

do 215

end 209–210

end (in case statement) 212

false 106–107, 245–246

module 155

next 216

non-useability as variable
 names 178

not 209

parentheses after
 consecutive 209

proc deprecated because too
 similar to Proc.new 356

rescue 226

return 102

self 179

true 106, 245

until 218

while 216

yield 219–224

knowing what your code is
 actually doing 68

L

lambda (keyword) 80

creating anonymous func-
 tions with 355

- lambda (keyword) *(continued)*
 - supplying code block to and calling 355–356
 - layouts 54, 78, 456
 - default 54
 - defined 54
 - learning to do more in your code 77–85
 - legacy data, converting to ActiveRecord 85–89
 - libraries 21
 - LIFO (last in, first out) 157
 - lightTPD Web server 61
 - Linux 18, 74
 - kernel configuration file 74
 - literal constructors 234–235
 - cannot be redefined 237
 - colon (for symbol) 235
 - curly braces (for hashes) 235
 - ellipsis (for ranges) 235
 - for arrays ([]) 235, 279
 - for regular expressions (//) 315
 - forward slashes (for regular expression) 235
 - more than one meaning for some constructs 235
 - overloading of notation 235
 - quotation marks (for string) 235
 - unambiguous in context 235
 - literals 100
 - load path, includes current directory 461
 - load vs. require 15
 - local variables
 - as temporary storage, vs. direct use of method return values 453
 - bareword appearance 119–120
 - creating through assignment 116
 - creating through method arguments 116
 - encapsulation with 453
 - legal characters for names of 119
 - local variables, 115–120
 - log files 59
 - loops 215–218
 - conditional, with while and until 216–218
 - terminating with break 216
 - through list of values with for 218
 - unconditional 215
 - until keyword 215
 - while and until as modifiers 218
 - while and until at end of loop 217
 - while keyword 215
 - luggage as example of stacklike behavior 161
- M**
-
- main 203
 - map operations 394
 - Masatoshi, Seki 29
 - match operation, treating as true/false 319
 - MatchData class 319–323
 - as example of dangers of changing core Ruby functionality 366–367
 - captures array 321
 - data about pattern matched stored in 321
 - end method 323
 - post_match method 323
 - pre_match method 323
 - string method 321
 - Matrix class (standard library) 289
 - Matsumoto, Yukihiro 110, 364
 - messages
 - as part of two-phase process 100
 - forwarding unrecognized to a designated object 360
 - sending to objects 99
 - sending with Kernel#send method vs. dot operator 111–112
 - sent to object via reference in variable 118
 - message-sending 7
 - metaphors for computer programs 178
 - method 294
 - method access rules 178–202
 - facilitating 201
 - in relation to 178
 - use in Rails controllers 179
 - method arguments 100
 - considerations of order of 302
 - default values for 113–114
 - importance of correct order of 114–115, 131
 - last starred (*) as argument 113
 - mini_link_to example of hash form of 302
 - pros and cons of using special hash construct 302
 - Rails convention favoring special hash construct in 303
 - required 112–113
 - summary of permutations 114
 - supplying correct number of 112
 - using special hash construct 301
 - variable number (using *) 113
 - method calls 207, 238
 - basic scenario always the same 238
 - introduction to 7–8
 - return value generated by every 244
 - sometimes include a code block 238
 - sometimes include arguments 238
 - that modify their receivers 238
 - method definitions 99
 - as contextual change 178
 - scope 116
 - method lists 254–255
 - method lookup order 163, 238
 - convergence of several language structure aspects 163
 - in cases of mixed classes and modules 155

- method lookup path
 - bypassing with method unbinding and binding 358
 - effect of module inclusion on 363
 - first match 358
 - method_missing, large role in Rails 361
 - methods
 - + (string concatenation) 84
 - arguments 101
 - as objects 357–359
 - automatically available via classes 122
 - binding to an object 357
 - calling syntax 7
 - conservatism about adding singleton 152
 - definitions 99
 - efficiency of not changing receiver 239
 - ending with question-mark 106
 - example of overriding 124
 - grabbing with instance_method method 357
 - input to, vs. keyboard input 101
 - listing an object's 234
 - overriding 124
 - real-world object modeling 103
 - reasons for handling as objects 358
 - redefining 124
 - return values 101–102, 238
 - risk of confusion from changing receiver 239
 - setter (=terminated) 130–136
 - setters 153
 - that change their receivers 234, 238–242
 - unbound 357
 - mix-ins *See* modules
 - Model/View/Controller (MVC) framework concept 36–38, 58–59, 65
 - actions and view tightly linked 40
 - breakdown reflected in Rails application directory structure 37
 - controller 36
 - division of labor 37
 - model 36
 - Rails implementation of 38–40
 - reconceived as MCV 36, 41
 - separation of programming concerns 37
 - three-part structure applicable generally to Rails understanding 37
 - traditional input/calculation/output model 37
 - view 36
 - models (ActiveRecord)
 - adding functionality to 82–85
 - as incarnation of application 82
 - as Ruby classes 372
 - born with 180 instance methods 374
 - class methods 375
 - correspondence to database tables 82
 - criteria of existence 375
 - database record creation as super-charged 375
 - engineered to have most of the functionality they need 396
 - find method 51, 399
 - find_by_* method 444
 - open-ended programmatic enhancement 84
 - openness to adding of any method 404
 - pre-defined callbacks 83
 - Ruby object and database record varying independently 375
 - summary of create/delete and related methods 376
 - two lives of instances 374
 - versatile even without custom code 393
 - See also* programmatic enhancement of ActiveRecord models
 - modularization, Rails as source of examples 173
 - modules
 - and code reuse 155–157, 161
 - and program design 155, 176
 - and Rails framework design 155
 - cannot be instantiated 155–156, 158
 - closely related to classes 155
 - creating 155–163
 - in Rails boilerplate code 175
 - in Rails source code 173–176
 - mixing into classes 158–160
 - mix-ins vs. inheritance 171–173
 - querying for instance and/or class methods 253
 - renaming (wrapping) methods to suit sub-domain 162
 - vs. classes 155
 - vs. classes in program design 160
 - writing your own 155
 - musical work, determining country of 397–398
 - MySQL 42, 45
 - sample command for populating database 47
 - sample console session for initializing databases 45
- ## N
-
- nil 83, 246–247, 249
 - as an object 250
 - as default value for container elements 250
 - as default value for instance variables 250
 - as method return value 107
 - Boolean value of false 249
 - equipped with 250
 - on Regexp#match failure 316, 321
 - one of two false objects 107, 249
 - only instance of NilClass 249
 - represented by the empty string 408

nil (*continued*)
 represents absence and state
 of being undetermined 250
 numerical objects 253, 270–272

O

Object (built-in class) 98, 122
 object state 127, 130
 altering 127
 at initialization 128–130
 changing dynamically 131
 changing with =terminated
 methods 132
 different shades of meaning
 for different classes 241
 reading 127, 129
 Object, highest class in
 hierarchy 155, 163
 object_id (method) 109–110
 object-oriented
 programming 97–98
 and real-world entities
 97–102
 model 96
 program design 98
 simultaneously simple and
 obscure 98
 vs. procedural 98
 objects 7–8, 97
 actions 97
 adding capabilities to 99
 addressing in the second
 person 178
 as agents and proxies for pro-
 grammer intentions 98
 as instances of classes 122
 as receivers of messages
 7, 100
 batch vs. individual
 creation 104
 Boolean values 107
 calling methods vs. sending
 messages 100
 can be different even if con-
 tents are the same 109
 centrality to Ruby 126
 creating 122
 default string representation
 of 243
 equality of 110

 exhibiting different
 capabilities 97
 id number 161
 id number associated with
 each 109
 innate behaviors 99, 108–112
 instantiation 122, 374
 listing methods of 253–255
 nature vs. nurture in 151–153
 not constrained by their
 class 151
 responding to messages 100
 sending messages to 7, 99
 teaching new behaviors 122
 opus numbers
 determining numerical
 content with match
 operation 407
 not always just numeric 407
 prettification of 404
 representing as 407
 special catalog designation
 instead of 407
 output
 file 13
 p 6, 331
 print 6
 puts 6

P

parentheses
 around method arguments 7
 conventionally not used in
 many Rails constructs
 71, 101
 favored by most Rubyists even
 when optional 101
 optional around method
 arguments 71, 101, 133
 partials (ActionPack partial
 templates) 424
 basic deployment 425–435
 for customer favorites 452
 for customer login 439
 for customer signup 439
 shopping cart 446, 449
 perception of the world
 and programming
 languages 97
 Perl 10, 97, 243, 314
 playing cards, as example of
 array-like domain 289
 pop (Array method) 158
 PostgreSQL 42, 45
 prettification of strings
 can involve more than just
 concatenation 404
 private (method access level)
 for security 201
 in controller files 199–201
 inheritance and 202
 Proc objects 351–356
 arguments and 353
 as closures 352
 behavior of possibly in flux in
 future versions of Ruby 355
 body taken from code block
 supplied to constructor 352
 calling multi-parameter Procs
 with too few arguments 353
 calling multi-parameter Procs
 with too many
 arguments 354
 code not executed until
 called 351
 context of creation and 353
 converting to code block 356
 difference between creating
 with new and with lambda
 keyword 355
 different handling of objects
 than methods 353
 local variables still in scope in
 block 352
 returning from call to 355
 warning for calling with too
 many arguments 353
 profiler 28
 program files 5
 creating first 8
 creating separate directory for
 samples 8
 importance of multiple in
 Ruby and Rails 15
 more than one per
 program 14–15
 naming conventions 5, 9
 programmatic enhancement of
 ActiveRecord models
 adding power and
 versatility 393

- programmatic enhancement of ActiveRecord models *(continued)*
 - advantages of programmatic vs. database solutions to some problems 413
 - class methods 419–421
 - consistency of syntax between built-in methods and added methods 395
 - different ways to determine country of musical work 398
 - example of 395
 - examples as pointers to kinds of things one can do 418
 - explained 393
 - full method status of 395
 - hard 404–419
 - keeping vs. discarding duplicate entries in collections 399
 - not to be done haphazardly 393
 - soft/hard distinction applied to class methods 419
 - soft/hard distinction as aid to choosing enhancements 396
 - soft/hard distinction as way of organizing survey of 396
 - string 404–409
 - too much as sign of need to redesign 396
 - vs. static storage of information in database 416
 - whole_name method (Composer) 395
 - programming freedom encouraged by Rails framework design 73
 - programs, getting your bearings in 178
 - protected (method access level) 201
 - callable if self is instance of receiver's class 201
 - mainly used for pairs of objects 201
 - variant of private 201–202
 - push (Array method) 158
 - puts, adding newline 10
- Q**
-
- queues 157
 - quoting mechanisms 259
 - %q and %Q 259
 - choice of delimiters for %q and %Q strings 260
- R**
-
- R4RMusic (first version)
 - actions 50
 - breakdown reflected in Rails application directory structure 38
 - composer/show 56–57
 - controller files 50
 - creating directory for with rails utility 37
 - creating the databases for 45
 - designing database tables for 46
 - edition/show 56–57
 - entities (models) 43
 - logic of modeling work-WORK and EDITION edition separately 43
 - main/welcome 50–51
 - mapping entities into 44
 - model files (ActiveRecord) for 48
 - modeling the domain 43–50
 - sample SQL data for 49
 - specifying actions for 50–52
 - summary of controller actions 50
 - work/show 53, 56
 - R4RMusic (sample application) 40
 - R4RMusic (second version)
 - actions available without login 439
 - ActiveRecord model design 372
 - add_to_cart action 449
 - adding new entities 372
 - all_periods class method (work model) 420
 - balance method (customer model) 417
 - balance method (customer model), inject version 418
 - basing one method on another 400
 - calculating customer's unpaid balance 417
 - century method (work model) 409, 412
 - check_out action 449–450
 - check_out method (Customer model) 450
 - check_out method (customer model), first version 418
 - check_out method (customer model), second version 418
 - check_out view 450
 - check-out process split between model and controller phases 418
 - completing purchases included in stub form 449
 - composer model 394
 - Composer#editions 401–402
 - composer_rankings method (customer model) 415
 - composers relatively inactive part of domain 401
 - controller enhancement 423
 - country method (Work) 398
 - customer 450
 - customer controller 446
 - customer favorites derived from rankings 450–452
 - customer login 438
 - Customer#edition_history 400
 - Customer#open_orders 399
 - Customer#work_history 400
 - Customer#works_on_order 400
 - customer/login 438, 440
 - customer/signup 438, 444
 - database design 372
 - domain modeling 372
 - dynamic determination of favorites target 451

- R4RMusic (second version)
 - (*continued*)
 - edition model as point of entry to Rails source code exploration 460
 - edition/show view 448
 - edition_history method (Customer) 400
 - editions method (Composer) 394, 401–402
 - editions_on_order method (Customer) 400
 - error reporting 443
 - favorites method 451
 - first iteration of customer rankings methods 414
 - generic 451
 - helper methods 427
 - hypothetical Period class 410
 - hypothetical weighted_instruments method 416
 - instrument_rankings method (customer model) 415
 - learning tool you can modify at will 453
 - levels of access 442
 - link_to_composer (helper method) 425
 - link_to_edition (helper method) 428
 - link_to_edition_title (helper method) 428
 - link_to_instrument (helper method) 428
 - link_to_work (helper method) 428
 - logout button in layout 445
 - methods built on other methods 401
 - music store vs. library 398
 - navigation bar added 445
 - nice_instruments method 405
 - nice_instruments method (work model) 408, 416
 - nice_opus method (work model) 407–408
 - nice_title method (edition model) 409
 - nice_title method (work model) 408
- of_works class method (edition model) 419
- open_orders method (customer model) 418
- open_orders method (Customer) 399
- ordered_by (Work) 398
- period method (work model) 412
- PERIODS hash (work model) 412
- prettyfied title for editions 409
- pretty title for work model 407
- publishers method (Composer) 401
- publishers method (Work) 398
- revising existing entities 372
- revising SQL table definitions 372
- sales_rankings class method (composer model) 421
- sales_rankings class method (work model) 420
- second iteration of customer ranking methods 415
- security concerns 442
- session defined by login action 439
- showing customer 442, 453
- signup action 444
- summary of helper methods and controllers 427
- summary of new actions 423
- summary of partials used in 435
- two_dec (helper method) 428
- view enhancement 423
- view_cart 446–447
- whole_name method (Composer) 394
- Work model 384
- Work#ordered_by 398
- works_on_order method (Customer) 400
- Rails 372
 - dependence of on Ruby's dynamism 338
- R4RMusic sample application 41–59
- rails (command-line utility) 37
- Rails API documentation consulting to understand source code 455
- RDoc format of 456
- Rails applications developing 35–36
- maintaining session continuity in 439
- Rails session lifecycle 59–65
- Rails source code
 - ActiveRecord subdirectories 457
 - advantages of becoming familiar with 68
 - belongs_to as sample 457
 - documentation not the only beneficial way to explore 456
 - exploring the source-code tree 456
 - importance of multiple files in 15
 - judgements required when 458
 - role of method_missing in 361
 - silo organization 457
 - transliteration of belongs_to into simple structure 463
- Rails special variables
 - @content_for_layout 54
 - @page_title 443
- Rainbow class as example of enumerability 305
- ranges of years for specifying musical periods 411
- RCRchive (Ruby Change Request Archive) 79, 81, 200
 - controller file for User 200
 - helper method example 81
 - sorting example 79
 - use of private methods for security 201
- RDoc (Ruby Documentation utility) 29
- RDoc markup in Ruby source files 29

- receivers changed during
 - method calls 234
- references 109, 118
 - and changing objects 118
 - as quasi-pointers 118
 - assigned from one variable to another 117
 - stored in variables 117–118
- reflection 111
 - filtering method lists with 254
 - instance_methods
 - method 254
 - listing an object's
 - methods 234
 - methods method 254
 - object_id (method) 109–110
 - private_methods method 255
 - protected_methods
 - method 255
 - public_methods method 255
 - respond_to? 110
 - singleton_methods
 - method 255
- regular expressions 312
 - \$ (end-of-line anchor) 327
 - \$1, \$2, etc. 320
 - * quantifier (zero or more) 324
 - + quantifier (one or more) 325
 - ? quantifier (zero or one) 324
 - \d ([0-9] character class) 318
 - \D (negation of \d class) 318
 - \S (negation of \s class) 318
 - \s (whitespace class) 318
 - \w ([0-9A-Za-z_] character class) 318
 - \W (negation of \w class) 318
 - \z (end-of-string anchor) 327
 - \Z (end-of-string anchor, discounting final newline character) 327
 - ^ (beginning-of-line anchor) 327
 - ^ (negation token in character classes) 318
 - { (repetition specifier) 326
 - (word-boundary anchor) 327
 - A (beginning-of-string anchor) 327
- abbreviations for common
 - character classes 318
- anchors 326–328
 - and Enumerable#grep 334
 - and String#gsub 334
 - and String#sub 333
- as arguments to built-in methods 313
- as representations of
 - patterns 314
- assertions 326
- basics of pattern matching 315
- begin method of MatchData class 323
- building patterns 316–319
- capture variables matching parentheses left to right 320
- character classes 316–317
- character classes are longer than what they match 319
- character ranges in character classes 318
- comma-separated fields
 - example 319
- common methods that use 331–335
- constraints on components of 315
- consumption of a character in string 326
- converting to/from strings 329–331
- difference between strings and 314–315, 329
- escaping special characters in interpolated strings 330
- escaping special characters with backslash 317
- general remarks on 313–314
- greediness 323
- greediness of quantifiers 325
- i (case-insensitivity)
 - modifier 328
- importance of learning about 313
- literal characters in 316
- literal characters match themselves in 317
- literal constructor (//) 315
- lookahead assertions 328
- m modifier (include newline in .) 328
- match method 315
- match success and failure 321–323
- MatchData object return on
 - successful match 320
- modifiers 328–329
- multiline mode (m modifier) 328
- negation in character classes 318
- negative assertions 328
- nil returned by unsuccessful match operation 321
- non-greedy quantifiers 325
- pattern-matching and 313
- patterns expressed in plain language 314
- positive assertions 328
- post_match method of MatchData 323
- pre_match method of MatchData 323
- quantifiers 323
- range of built-in methods that use them 313
- Regexp.escape class
 - method 330
- reputation as opaque and unmaintainable 313
- scan operations with 332
- special characters 317
- special characters in interpolated strings remain special 329
- special variables for parenthetical captures 320
- splitting string with 332
- string interpolation
 - inside 329
- string representation of 331
- techniques 323–331
- two ways of getting captures from MatchData object 322
- useful in converting legacy data 314
- viewing as strings with inspect method 331

- regular expressions (*continued*)
 - wildcard character (.)
 - 316–317
 - wildcard character
 - matches 317
 - wizardry not necessarily indis-
pensable to Rails applica-
tion development 314
 - writing 314–319
 - zero-width assertions 326, 328
 - relational databases 96
 - repetition
 - avoiding by extracting com-
mon code into separate
method 415
 - avoiding in code 137
 - in customer rankings
methods 415
 - require 14, 22
 - foundation of much of Ruby's
power 21
 - prominence in
 - active_record.rb source
file 457
 - vs. load 15
 - rescue blocks
 - begin/end delimited 226
 - capturing exception to a
variable in 228
 - inside method
 - definitions 226, 228
 - respond_to? (method) 109–111
 - return (keyword) 102
 - ri (Ruby Index utility) 29
 - routes (ActionController) 58
 - configuration file
(routes.rb) 58
 - Ruby
 - as administrative and organi-
zational tool 68–69, 85–90
 - as host language for DSLs 69
 - basic language literacy 5
 - bootstrapping knowledge of 5
 - breadth of standard library 21
 - breaking 372
 - components of programming
environment 23
 - core language vs. standard
library 21
 - dynamic nature of 338
 - facility with as aid to Rails
programming 438
 - high power-to-lines-of-code
ratio 105
 - how it helps the Rails
developer 68–69
 - installation 5–6, 25–27
 - installing source with package
manager 24
 - learning process 234
 - programming
 - environment 24–31
 - source code 24–25
 - standard library 21–22
 - standard tools and
applications 27–31
 - syntax checking vs. semantic
policing 133
 - syntax, relatively
uncomplicated 70
 - techniques you should under-
stand but not necessarily
use 359
 - terminological
conventions 136
 - view from 4
 - writing a program 4–15
 - ruby (interpreter)
 - as program others are fed
to 16
 - building 25
 - invocation 15–21
 - verbose mode 18
 - warning mode 17
 - ruby (Ruby interpreter) 4, 9
 - Ruby Application Archive
(RAA) 23
 - Ruby on Rails
 - application directory
structure 37
 - application naming 38
 - application operations 34
 - as domain-specific
language 70–73, 77
 - as Ruby environment 73, 85
 - as simultaneous deployment
of multiple libraries 40
 - built-in support for
customization 78, 82–83
 - code as part of application
code 34
 - coding conventions 72–73, 75
 - configuration 69, 71, 74
 - controllers directory 38
 - designed for use 35
 - developing with vs. developing
without 35
 - dispatcher scripts 60–62,
65, 459
 - domain vs. domain of
applications 71
 - environment.rb configura-
tion file 459
 - framework overview 34
 - framework source code 174
 - helper files 78
 - helper methods 82
 - layered philosophically on top
of Ruby 136
 - legacy data and 85
 - made up of three main pro-
gramming libraries 38
 - medium-level overview 34
 - method-call 301
 - models directory 38
 - modularity of framework
design 155
 - open-ended programmer
freedom 78, 80
 - pre-determined aspects of
applications 78
 - range of possible
applications 34
 - source code directory
structure 39–40
 - support libraries 40
 - three levels of development
freedom 77
 - views directory 38
 - RubyForge 23
 - RubyGems (packaging
system) 38, 473–475
- S**
- scalar objects 258
 - scope 173, 178
 - new local inside method
definition body 352
 - new, in method
definitions 116

- security for checking incoming data 444
- self (default object)
 - an object rather than a concept 178
 - as first-person of the program 178
 - changes during program execution 178
 - how Ruby determines it 179
 - privileges of being 179
 - rules for determining which object is 179
 - unique at any given time 179
- send 109
 - as explicit message-sending technique 71, 111–112
- session variable
 - persistence of data across actions 440
 - storing customer ID in 440
- session variable (@session) 440
- session, different definitions for different sites 439
- Set class (standard library) 289
- setter (=terminated) methods
 - abuse of syntax 133
 - as data filters 134
 - in ActiveRecord model files 135
- SHA1 encryption 444
- shopping cart
 - adding items 449
 - not an object but a view of other objects 446
- singleton methods 124
- site-ruby directory 26
- soft enhancements, Ruby vs. SQL 401–403
- sorting techniques
 - array most common container object for sort operations 307
 - in relation to Comparable module 309
 - not necessary to mix Enumerable into class of objects to be sorted 307
 - objects that know how to be sorted 308
 - on objects of different, incomparable classes 310
 - on objects with no method 310
 - role of method in 307, 309
 - sort method of
 - Enumerable 307
 - sort_by method of
 - Enumerable 80, 307, 310
 - sorting against a fixed non-alphabetical array 405
 - sorting two objects extrapolated to entire collection sort 308
 - use of code block in 309
 - use of container objects for 307
 - using a code block in 310–311
- special variables, `__FILE__` 461
- SQL 40, 42, 49, 83
 - controllers creating automatically 64
 - fragments of as arguments to ActiveRecord find methods 291
 - id field as primary key 46
 - NULL equivalent to Ruby nil 83
 - vs. pure Ruby for collection searching 292
- SQLite 42, 45
- stacklike (sample module) 157–158
- stacklike, simple implementation 157
- stacks 157
 - implemented via arrays 158
 - implementing in module vs. class 160–163
 - plates as example of 157
 - real-world examples of 157, 161
 - vs. arrays 158
- standard library 21–22
- string interpolation 105
 - #{} operator 106
 - as string-combining technique 262
 - doesn't work inside single-quoted strings 258
 - inside regular expressions 329
 - learning by using irb 259
- strings
 - + method 261
 - << method 261
 - as enumerables 306–307
 - basics 258–260
 - capitalize method 263
 - capitalizing 239
 - chomp method 263
 - chop method 263
 - combining two or more 261
 - conversion of to integers with `to_i` 243
 - disabling interpolation by escaping # character 259
 - downcase method 263
 - each method iterates through lines rather than characters 306
 - each_byte method 306
 - escaping backslashes in 259
 - escaping characters inside single vs. double quotation marks 259
 - escaping quotation marks 259
 - gsub and gsub! methods 333–334
 - hypothetical method for character-wise iteration 307
 - include Comparable module methods 253
 - literals (with quotation marks) 258
 - lstrip method 263
 - massaging 263
 - match method 315
 - multiline with newline character 306
 - multiplication of with * 243
 - one of two ways of representing text 258
 - operations 260
 - replacing contents of 239, 262
 - reverse method 263
 - rstrip method 263
 - scan method 332
 - single vs. double quotation marks 258
 - split method 332–333
 - strip method 263

strings (*continued*)

- sub and sub! methods 333–334
- substitution methods 333–334
- succ method 410
- swapcase method 263
- to_s method of return receiver 242
- uppercase method 263
- versatility of 260

subclasses 148

submatches, capturing with parentheses 319

subsumed under ActionController 39

- ActionController 39
- ActionView 39

superclasses 148

symbols 71

- and configuration 73–75
- as arguments 72
- as hash keys 73–74
- as method arguments 73
- one of two ways of representing text 258
- preliminary characterization of 138

syntactic sugar 132–133, 135, 212–213, 225, 234, 255, 272, 281, 294

- and Ruby philosophy 237
- arithmetic operators and 237
- categories of automatic 237
- for + method 236
- for setter (=terminated) methods 132
- making method calls look like operators 237
- methods that always exhibit 236
- potential abuse of 237
- recurrent 236–238
- special treatment of = (equal sign) 237
- summary of automatic 236
- summary of operator-style notation 236

syntax errors 9

syntax, checking accuracy of 16

T

temperature conversion (code example) iterator version 222

temperature converter (code example) 218

temperature converter (sample program) 101, 142

- adding class methods to 142

text editor, for writing program files 5, 8

Thomas, Dave 29

tickets to events (code example) 103–108

Time (Ruby class) 125

- strftime method 125
- time.rb extension file 125
- xmlschema (extension method from time.rb) 126

time and date objects 272–275

- classes pertaining to 272
- format strings for controlling output 274

to_* (conversion)

- methods 242–244
- built-in 242
- every object has to_s 242
- not all objects have all 242
- to_a (to array) 242
- to_f (to float) 242
- to_i (to integer) 242
- to_s (to string) 242
- to_s as most common 242
- to_s automatically called by puts 244
- writing a custom default to_s for a class 243

top-level

- methods 203–204
- methods as private instance methods of Object 203
- Rails scripts written at 203
- scripts vs. class/module-based programs 203

top-level context 179

two lives of ActiveRecord model object

- as handle for manipulating database record 375
- as Ruby object 375

U

unification

- of components of Time class 126
- possibility of for date and time libraries 273

URLs

- as determinant of controller/action sequence 150
- rewriting by Rails routing subsystem 459

V

variable assignment 6, 98, 117–119

- evaluates to right-hand side 20

variables 96

- distinct from object referred to 109
- local 115–120
- reassigning to existing 118
- visibility 178

view designing 53–58

views, fulfillment 65

W

warnings

- for reassigning to a constant 123
- for wrong number of arguments to a block 353

Web site for Ruby for Rails 438

WEBrick 58–60

- server script 59

Weirich, Jim 53

word processor for writing program files 5

work (musical) period

- analogous to other arts 410
- calculating 409–413
- creating descriptions based on year and country of origin 410
- date and country pairs stored in hash 411

work (musical) period
(continued)
 descriptive form of 410
 finding all represented 420
 hypothetical Period class 410
 ranking sales of all 420
 work (musical) title
 components of 408
 Time class 272
 www.ruby-lang.org 24

X

XML Schema dateTime
 representation 126

Y

YAML 75–77, 85
 and converting legacy data 87
 data serialization with 75

history of acronym 75
 load (deserialization)
 method 75
 loading library 75
 preservation of data classes
 during serialization 76
 sample 76
 serialization of data to string
 form 88
 to_yaml method 75
 use of by Rails 76