

contents

目次

本書の刊行に寄せて.....	iii
まえがき.....	iv

第1章 本書を読み進める前に 1

1.1 イントロダクション.....	2
1.1.1 この章で説明すること.....	2
1.2 本書の概要.....	2
1.2.1 対象となる読者.....	2
1.2.2 本書で説明する内容と説明しない内容.....	3
1.2.3 Rails アプリの開発にも本書は役に立つか?.....	4
1.2.4 本書の特徴と効果的な学習方法.....	4
1.2.5 対象となるRubyのバージョン.....	5
1.3 Rubyについて.....	6
1.3.1 Ruby ってどんなプログラミング言語?.....	6
1.3.2 Rubyの処理系.....	6
1.3.3 Rubyのライセンス.....	6
1.4 Rubyのインストール.....	7
1.4.1 Mac/Linuxの場合.....	8
1.4.2 Windowsの場合.....	8
1.4.3 その他の方法.....	9
1.4.4 サンプルコードの動作確認環境について.....	9
1.5 エディタ / IDE について.....	10
1.5.1 エディタやIDEで不可欠な機能や設定.....	10
1.6 Rubyを動かしてみる.....	11
1.6.1 本書のサンプルコードとその表示例について.....	12
1.6.2 ファイルに保存したRubyプログラムを実行する.....	14
1.7 本書のサンプルコードがうまく動かない場合.....	15
1.8 この章のまとめ.....	16

第2章 Rubyの基礎を理解する 17

2.1 イントロダクション.....	18
2.1.1 この章の例題: FizzBuzzプログラム.....	18
2.1.2 FizzBuzzプログラムの実行例.....	18
2.1.3 この章で学ぶこと.....	18
2.2 Rubyに関する基礎知識.....	19
2.2.1 すべてがオブジェクト.....	19
2.2.2 メソッド呼び出し.....	19
2.2.3 文の区切り.....	20
2.2.4 コメント.....	21
2.2.5 識別子と予約語.....	21
2.2.6 空白文字.....	21
2.2.7 リテラル.....	22

2.2.8	変数 (ローカル変数) の宣言と代入	22
2.3	文字列	24
2.3.1	シングルクォートとダブルクォート	24
2.3.2	文字列の比較	26
2.4	数値	27
2.4.1	演算子による値の比較	28
2.4.2	演算子の優先順位	28
2.4.3	変数に格納された数値の増減	29
2.5	真偽値と条件分岐	31
2.5.1	Rubyの真偽値	31
2.5.2	論理演算子	32
2.5.3	if文	33
2.6	メソッドの定義	36
2.6.1	メソッドの戻り値	37
2.6.2	メソッド定義における引数の()	38
2.7	例題: FizzBuzzプログラムを作成する	38
2.7.1	作業用のディレクトリとファイルを準備する	38
2.7.2	一番簡単なプログラムで動作確認する	39
2.7.3	fizz_buzzメソッドを作成する	39
2.8	文字列についてもっと詳しく	42
2.8.1	文字列はStringクラスのオブジェクト	42
2.8.2	%記法で文字列を作る	42
2.8.3	ヒアドキュメント (行指向文字列リテラル)	43
2.8.4	フォーマットを指定して文字列を作成する	45
2.8.5	その他、文字列作成のいろいろ	46
2.8.6	文字と文字列の違いはない	46
2.9	数値についてもっと詳しく	47
2.9.1	10進数以外の整数リテラル	47
2.9.2	ビット演算	47
2.9.3	指数表現	47
2.9.4	数値クラスのあれこれ	48
2.10	真偽値と条件分岐についてもっと詳しく	49
2.10.1	&&や の戻り値と評価を終了するタイミング	49
2.10.2	優先順位が低いand、or、not	50
2.10.3	unless文	52
2.10.4	case文	53
2.10.5	条件演算子 (三項演算子)	55
2.11	メソッド定義についてもっと詳しく	56
2.11.1	デフォルト値付きの引数	56
2.11.2	?で終わるメソッド	57
2.11.3	!で終わるメソッド	58
2.12	その他の基礎知識	59
2.12.1	ガベージコレクション (GC)	59
2.12.2	エイリアスメソッド	59
2.12.3	式 (Expression) と文 (Statement)	60
2.12.4	擬似変数	60
2.12.5	参照の概念を理解する	61
2.12.6	組み込みライブラリ、標準ライブラリ、gem	63
2.12.7	require	63
2.12.8	load	64
2.12.9	require_relative	64
2.12.10	putsメソッド、printメソッド、pメソッド	65

2.13	この章のまとめ	67
第3章	テストを自動化する	69
3.1	イントロダクション	70
3.1.1	この章で学ぶこと	70
3.1.2	「プログラムの三大美徳」	70
3.2	Minitestの基本	71
3.2.1	テストコードのひな形	72
3.2.2	本書で使用するMinitestの検証メソッド	73
3.2.3	テストコードの実行と結果の確認	73
3.2.4	テストが失敗した場合の実行結果	74
3.2.5	実行中にエラーが発生した場合の実行結果	75
3.3	FizzBuzzプログラムのテスト自動化	76
3.3.1	putsメソッドをテストコードに置き換える	76
3.3.2	プログラム本体とテストコードを分離する	80
3.4	この章のまとめ	82
第4章	配列や繰り返し処理を理解する	85
4.1	イントロダクション	86
4.1.1	この章の例題：RGBカラー変換プログラム	86
4.1.2	RGBカラー変換プログラムの実行例	86
4.1.3	この章で学ぶこと	86
4.2	配列	87
4.2.1	要素の変更、追加、削除	88
4.2.2	配列を使った多重代入	89
4.3	ブロック	90
4.3.1	参考：Javaの繰り返し処理	90
4.3.2	Rubyの繰り返し処理	91
4.3.3	配列の要素を削除する条件を自由に指定する	92
4.3.4	ブロック引数とブロック内の変数	93
4.3.5	do ... end と {}	95
4.4	ブロックを使う配列のメソッド	96
4.4.1	map/collect	96
4.4.2	select/find_all/reject	96
4.4.3	find/detect	97
4.4.4	inject/reduce	97
4.4.5	&とシンボルを使ってもっと簡潔に書く	98
4.5	範囲 (Range)	99
4.5.1	配列や文字列の一部を抜き出す	100
4.5.2	n以上m以下、n以上m未満の判定をする	101
4.5.3	case文で使う	101
4.5.4	値が連続する配列を作成する	102
4.5.5	繰り返し処理を行う	102
4.6	例題：RGB変換プログラムを作成する	103
4.6.1	to_hexメソッドを作成する	103
4.6.2	to_hexメソッドをリファクタリングする	106
4.6.3	to_intsメソッドを作成する	108
4.6.4	to_intsメソッドをリファクタリングする	111
4.6.5	to_intsメソッドをリファクタリングする (上級編)	112

4.7	配列についてもっと詳しく	115
4.7.1	さまざまな要素の取得方法.....	115
4.7.2	さまざまな要素の変更方法.....	116
4.7.3	配列の連結.....	116
4.7.4	配列の和集合、差集合、積集合.....	117
4.7.5	多重代入で残りの全要素を配列として受け取る.....	118
4.7.6	1つの配列を複数の引数として展開する.....	119
4.7.7	メソッドの可変長引数.....	119
4.7.8	*で配列同士を非破壊的に連結する.....	120
4.7.9	==で等しい配列かどうか判断する.....	120
4.7.10	%記法で文字列の配列を簡潔に作る.....	120
4.7.11	文字列を配列に変換する.....	121
4.7.12	配列に初期値を設定する.....	122
4.7.13	配列に初期値を設定する場合の注意点.....	122
4.7.14	ミュータブル? イミュータブル?.....	124
4.8	ブロックについてもっと詳しく	125
4.8.1	添え字付きの繰り返し処理.....	125
4.8.2	with_indexメソッドを使った添え字付きの繰り返し処理.....	125
4.8.3	添え字を0以外の数値から開始させる.....	126
4.8.4	配列がブロック引数に渡される場合.....	127
4.8.5	ブロックローカル変数.....	129
4.8.6	繰り返し処理以外でも使用されるブロック.....	130
4.8.7	do...endと{}の結合度の違い.....	130
4.8.8	ブロックを使うメソッドを定義する.....	131
4.9	さまざまな繰り返し処理	132
4.9.1	timesメソッド.....	132
4.9.2	uptoメソッドとdowntoメソッド.....	133
4.9.3	stepメソッド.....	133
4.9.4	while文とuntil文.....	133
4.9.5	for文.....	135
4.9.6	loopメソッド.....	136
4.10	繰り返し処理用の制御構造	138
4.10.1	break.....	138
4.10.2	throwとcatchを使った大域脱出.....	139
4.10.3	繰り返し処理で使うbreakとreturnの違い.....	141
4.10.4	next.....	142
4.10.5	redo.....	143
4.11	この章のまとめ	144

第5章 ハッシュやシンボルを理解する 145

5.1	イントロダクション	146
5.1.1	この章の例題：長さの単位変換プログラム.....	146
5.1.2	長さの単位変換プログラムの実行例.....	146
5.1.3	この章で学ぶこと.....	146
5.2	ハッシュ	147
5.2.1	要素の追加、変更、取得.....	148
5.2.2	ハッシュを使った繰り返し処理.....	149
5.2.3	ハッシュの同値比較、要素数の取得、要素の削除.....	149
5.3	シンボル	150
5.3.1	シンボルと文字列の違い.....	151
5.3.2	シンボルの特徴とおもな用途.....	152

5.4	続・ハッシュについて	153
5.4.1	ハッシュのキーにシンボルを使う.....	153
5.4.2	キーや値に異なるデータ型を混在させる.....	153
5.4.3	メソッドのキーワード引数とハッシュ.....	154
5.5	例題：長さの単位変換プログラムを作成する	156
5.5.1	テストコードを準備する.....	156
5.5.2	いろんな単位を変換できるようにする.....	158
5.5.3	convert_lengthメソッドを改善する.....	159
5.6	ハッシュについてもっと詳しく	162
5.6.1	ハッシュで使用頻度の高いメソッド.....	162
5.6.2	**でハッシュを展開させる.....	163
5.6.3	ハッシュを使った疑似キーワード引数.....	163
5.6.4	任意のキーワードを受け付ける**引数.....	164
5.6.5	メソッド呼び出し時の{}の省略.....	164
5.6.6	ハッシュリテラルの{}とブロックの{}.....	165
5.6.7	ハッシュから配列へ、配列からハッシュへ.....	166
5.6.8	ハッシュの初期値を理解する.....	167
5.7	シンボルについてもっと詳しく	169
5.7.1	シンボルを作成するさまざまな方法.....	169
5.7.2	%記法でシンボルやシンボルの配列を作成する.....	170
5.7.3	シンボルと文字列の関係.....	170
5.8	この章のまとめ	171

第6章 正規表現を理解する 177

6.1	イントロダクション	178
6.1.1	この章の例題：ハッシュ記法変換プログラム.....	178
6.1.2	ハッシュ記法変換プログラムの実行例.....	178
6.1.3	この章で学ぶこと.....	179
6.2	正規表現って何?	179
6.2.1	正規表現の便利さを知る.....	180
6.2.2	正規表現をゼロから学習するための参考資料.....	181
6.3	Rubyにおける正規表現オブジェクト	183
6.3.1	Rubularで視覚的にマッチする文字列を確認する.....	184
6.3.2	正規表現のキャプチャを利用する.....	185
6.3.3	キャプチャの結果に名前を付ける.....	187
6.3.4	正規表現と組み合わせると便利なStringクラスのメソッド.....	188
6.4	例題：Rubyのハッシュ記法を変換する	191
6.4.1	テストコードを準備する.....	191
6.4.2	ハッシュ記法変換プログラムを実装する.....	193
6.5	正規表現オブジェクトについてもっと詳しく	197
6.5.1	正規表現オブジェクトを作成するさまざまな方法.....	197
6.5.2	case文で正規表現を使う.....	197
6.5.3	正規表現オブジェクト作成時のオプション.....	198
6.5.4	組み込み変数でマッチの結果を取得する.....	199
6.5.5	Regexp.last_matchでマッチの結果を取得する.....	200
6.5.6	組み込み変数を書き換えないmatch?メソッド.....	200
6.6	この章のまとめ	201

7.1	イントロダクション	204
7.1.1	この章の例題：改札機プログラム	204
7.1.2	この章で学ぶこと	205
7.2	オブジェクト指向プログラミングの基礎知識	206
7.2.1	クラスを使う場合と使わない場合の比較	206
7.2.2	オブジェクト指向プログラミング関連の用語	209
7.3	クラスの定義	212
7.3.1	オブジェクトの作成と initialize メソッド	212
7.3.2	インスタンスメソッドの定義	213
7.3.3	インスタンス変数とアクセサメソッド	213
7.3.4	クラスメソッドの定義	218
7.3.5	定数	220
7.4	例題：改札機プログラムの作成	221
7.4.1	テストコードを準備する	222
7.4.2	必要なメソッドやクラスを仮実装する	223
7.4.3	運賃が足りているかどうかを判別する	226
7.4.4	テストコードのリファクタリング	229
7.4.5	残りのテストケースをテストする	231
7.5	self キーワード	233
7.5.1	self の付け忘れで不具合が発生するケース	234
7.5.2	クラスメソッドやクラス構文直下の self	235
7.5.3	クラスメソッドをインスタンスメソッドで呼び出す	237
7.6	クラスの継承	238
7.6.1	標準ライブラリの継承関係	239
7.6.2	デフォルトで継承される Object クラス	240
7.6.3	オブジェクトのクラスを確認する	240
7.6.4	ほかのクラスを継承したクラスを作る	241
7.6.5	super でスーパークラスのメソッドを呼び出す	242
7.6.6	メソッドのオーバーライド	244
7.6.7	クラスメソッドの継承	245
7.7	メソッドの公開レベル	246
7.7.1	public メソッド	246
7.7.2	private メソッド	246
7.7.3	private メソッドはサブクラスでも呼び出せる	248
7.7.4	クラスメソッドを private にしたい場合	249
7.7.5	private メソッドから先に定義する場合	250
7.7.6	あとからメソッドの公開レベルを変更する場合	251
7.7.7	protected メソッド	251
7.8	定数についてもっと詳しく	254
7.8.1	定数と再代入	255
7.8.2	定数はミュータブルなオブジェクトに注意する	256
7.9	さまざまな種類の変数	258
7.9.1	クラスインスタンス変数	258
7.9.2	クラス変数	260
7.9.3	グローバル変数と組み込み変数	262
7.10	クラス定義や Ruby の言語仕様に関する高度な話題	263
7.10.1	エイリアスメソッドの定義	263
7.10.2	メソッドの削除	264
7.10.3	ネストしたクラスの定義	264
7.10.4	演算子の挙動を独自に再定義する	265

7.10.5	等値を判断するメソッドや演算子を理解する	267
7.10.6	オープンクラスとモンキーパッチ	270
7.10.7	特異メソッド	272
7.10.8	クラスメソッドは特異メソッドの一種	273
7.10.9	ダックタイピング	275
7.11	この章のまとめ	280

第8章 モジュールを理解する 281

8.1	イントロダクション	282
8.1.1	この章の例題：deep_freezeメソッド	282
8.1.2	この章で学ぶこと	283
8.2	モジュールの概要	283
8.2.1	モジュールの用途	283
8.2.2	モジュールの定義	284
8.3	モジュールのミックスイン (includeとextend)	285
8.3.1	モジュールをクラスにincludeする	285
8.3.2	モジュールをextendする	288
8.4	例題：deep_freezeメソッドの作成	289
8.4.1	実装の方針を検討する	289
8.4.2	テストコードを準備する	290
8.4.3	deep_freezeメソッドを実装する	291
8.4.4	deep_freezeメソッドをハッシュに対応させる	293
8.5	ミックスインについてもっと詳しく	295
8.5.1	includeされたモジュールの有無を確認する	295
8.5.2	include先のメソッドを使うモジュール	296
8.5.3	Enumerableモジュール	297
8.5.4	Comparableモジュールと<=>演算子	297
8.5.5	Kernelモジュール	299
8.5.6	トップレベルはmainという名前のObject	300
8.5.7	クラスやモジュール自身もオブジェクト	301
8.5.8	モジュールとインスタンス変数	303
8.5.9	オブジェクトに直接ミックスインする	304
8.6	モジュールを利用した名前空間の作成	304
8.6.1	名前空間を分けて名前の衝突を防ぐ	304
8.6.2	名前空間でグループやカテゴリを分ける	306
8.6.3	ネストなしで名前空間付きのクラスを定義する	307
8.7	関数や定数を提供するモジュールの作成	309
8.7.1	モジュールに特異メソッドを定義する	309
8.7.2	module_functionメソッド	310
8.7.3	モジュールに定数を定義する	311
8.7.4	モジュール関数や定数を持つモジュールの例	311
8.8	状態を保持するモジュールの作成	312
8.9	モジュールに関する高度な話題	315
8.9.1	メソッド探索のルールを理解する	315
8.9.2	モジュールにほかのモジュールをincludeする	318
8.9.3	prependでモジュールをミックスインする	319
8.9.4	prependで既存メソッドを置き換える	320
8.9.5	有効範囲を限定できるrefinements	321
8.10	この章のまとめ	325

9.1	イントロダクション	328
9.1.1	この章の例題：正規表現チェッカープログラム.....	328
9.1.2	正規表現チェッカーの実行例.....	328
9.1.3	この章で学ぶこと	329
9.2	例外の捕捉	329
9.2.1	発生した例外を捕捉しない場合	329
9.2.2	例外を捕捉して処理を続行する場合	331
9.2.3	例外処理の流れ	332
9.2.4	例外オブジェクトから情報を取得する	334
9.2.5	クラスを指定して捕捉する例外を限定する	335
9.2.6	例外クラスの継承関係を理解する	336
9.2.7	継承関係と rescue 節の順番に注意する.....	337
9.2.8	例外発生時にもう一度処理をやりなおす retry	340
9.3	意図的に例外を発生させる	341
9.4	例外処理のベストプラクティス	342
9.4.1	安易に rescue を使わない.....	343
9.4.2	rescue したら情報を残す	343
9.4.3	例外処理の対象範囲と対象クラスを極力絞り込む.....	344
9.4.4	例外処理よりも条件分岐を使う	345
9.4.5	予期しない条件は異常終了させる	346
9.5	例題：正規表現チェッカープログラムの作成	348
9.5.1	テスト駆動開発をするかどうか	348
9.5.2	実装のフローチャートを考える	348
9.5.3	文字入力を受け付ける gets メソッド	349
9.5.4	実装を開始する	350
9.5.5	例外処理を組み込む	351
9.6	例外処理についてもっと詳しく	353
9.6.1	ensure.....	353
9.6.2	ensureの代わりにブロックを使う	353
9.6.3	例外処理の else.....	354
9.6.4	例外処理と戻り値	355
9.6.5	begin/end を省略する rescue 修飾子.....	357
9.6.6	!と \$@ に格納される例外情報.....	358
9.6.7	例外処理の begin/end を省略できるケース	358
9.6.8	rescue した例外を再度発生させる	360
9.6.9	独自の例外クラスを定義する	361
9.7	この章のまとめ	362

10.1	イントロダクション	364
10.1.1	この章の例題：ワードシンセサイザー	364
10.1.2	この章で学ぶこと	366
10.2	ブロックを利用するメソッドの定義と yield	366
10.2.1	yield を使ってブロックの処理を呼び出す	366
10.2.2	ブロックを引数として明示的に受け取る	369
10.3	Proc オブジェクト	373
10.3.1	Proc オブジェクトの基礎	373
10.3.2	Proc オブジェクトをブロックの代わりに渡す	374

10.3.3	Procオブジェクトを普通の引数として渡す	375
10.3.4	Proc.newとラムダの違い	376
10.3.5	Proc.newからラムダかを判断するlambda?メソッド	378
10.4	例題：ワードシンセサイザーの作成	378
10.4.1	エフェクトの実装方法を検討する	379
10.4.2	テストコードも2つに分ける	380
10.4.3	テストコードを準備する	380
10.4.4	リバースエフェクトを実装する	381
10.4.5	エコーエフェクトを実装する	382
10.4.6	ラウドエフェクトを実装する	383
10.4.7	WordSynthクラスの実装とテスト	384
10.5	Procオブジェクトについてもっと詳しく	388
10.5.1	Procオブジェクトを実行するさまざまな方法	388
10.5.2	&とto_procメソッド	389
10.5.3	Procオブジェクトとクロージャ	390
10.5.4	ややこしい2つめのProc.newとラムダの違い	391
10.6	この章のまとめ	393

第11章 Rubyのデバッグ技法を身につける 395

11.1	イントロダクション	396
11.1.1	この章で学ぶこと	396
11.2	バックトレースの読み方	396
11.3	よく発生する例外クラスとその原因	398
11.3.1	NameError	398
11.3.2	NoMethodError	399
11.3.3	TypeError	399
11.3.4	ArgumentError	400
11.3.5	ZeroDivisionError	400
11.3.6	SystemStackError	400
11.3.7	LoadError	400
11.3.8	SyntaxError (syntax error)	401
11.3.9	上記以外のRuby標準の例外クラス	401
11.3.10	Ruby標準ではない例外クラス	401
11.4	プログラムの途中経過を確認する	401
11.4.1	printデバッグ	402
11.4.2	tapでメソッドチェーンをデバッグする	403
11.4.3	ログにデバッグ情報を出力する	403
11.4.4	デバugg (Byebug) を使う	404
11.5	汎用的なトラブルシューティング方法	407
11.5.1	irb上で簡単なコードを動かしてみる	407
11.5.2	ログを調べる	407
11.5.3	公式ドキュメントを読む	407
11.5.4	issueを検索する	407
11.5.5	ライブラリのコードを読む	408
11.5.6	テストコードを書く	408
11.5.7	“警戒しながら” ネットの情報を参考にする	409
11.5.8	パソコンの前から離れる	409
11.5.9	誰かに聞く	410
11.6	この章のまとめ	410

第12章 Rubyに関するその他のトピック 411

12.1	イントロダクション	412
12.1.1	この章で学ぶこと	412
12.2	日付や時刻の扱い	412
12.3	ファイルやディレクトリの扱い	414
12.4	特定の形式のファイルを読み書きする	416
12.4.1	CSV	416
12.4.2	JSON	416
12.4.3	YAML	417
12.5	環境変数や起動時引数の取得	418
12.5.1	組み込み定数	418
12.6	eval、バッククオートリテラル、sendメソッド	420
12.7	Rake	422
12.7.1	Rakeの基本的な使い方	422
12.7.2	Rakeを使ったテストの一括実行	423
12.8	gemとBundler	425
12.8.1	gemのインストールと利用方法	425
12.8.2	Bundlerでgemの依存関係を管理する	427
12.8.3	Gemfileでgemのバージョンを指定する記号の意味	430
12.9	この章のまとめ	431

付録 Ruby on Railsの習得に向けた予備知識 433

A.1	イントロダクション	434
A.2	Railsの独自拡張になっている機能を理解する	435
A.3	フレームワークの変化の速さに追従する	437
A.4	アプリケーション設計に関する知識	438
A.5	Web技術に関する知識	438
A.6	データベースに関する知識	439
A.7	セキュリティに関する知識	440
A.8	テストの自動化に関する知識	440
A.9	GitやGitHubに関する知識	442
A.10	サーバや運用に関する知識	442
A.11	gemに関する知識と定期的なアップデート	443

あとがき 445

参考文献 447

索引 449

Column

Rubyのリリースサイクルについて	5
Windows環境とバックスラッシュ	11
irbで日本語は入力できますか？	13
ファイルエンコーディングとマジックコメント	15
シングルクオートかダブルクオートか	26
数値と文字列は暗黙的に変換されない	29
小数を使う場合は丸め誤差に注意	30
putsやpはグローバル関数？	67
Rubyのコーディングルール	67
require_relativeを使う場合	82
Minitest以外のテストフレームワーク	82
テスト駆動開発の開発サイクル	114
[]や<<を使った文字列の操作	125
ブロックのうしろに別のメソッドを続けて書く	131
配列をもっと上手に使いこなすために	132
繰り返し処理とEnumerableモジュール	137
ハッシュをもっと上手に使いこなすために	163
メソッド定義時の引数の順番	172
よく使われるイディオム (1) 条件分岐で変数に代入 / &. 演算子	173
よく使われるイディオム (2) =を使った自己代入	174
よく使われるイディオム (3) !!を使った真偽値の型変換	175
正規表現を使う場合は方言やRubyのバージョンに注意	201
irb上でクラス定義を繰り返す際の注意点	217
メソッド名の表記法について	220
RDocでAPIドキュメントを作成する	232
継承したら同名のインスタンス変数に注意	253
クラスの可視性について	265
オープンクラスやモンキーパッチの弊害	272
メソッドの有無を調べるrespond_to?	278
Rubyでメソッドのオーバーロード？	279
ハッシュのキーが文字列だった場合は自動的にfreezeされる	294
トップレベルの同名クラスを参照する	307
Singletonモジュールを使う	314
モジュールの用途は1つとは限らない	315
二重コロン (::) とドット (.) の違い	324
ensure節ではreturnを使わない	356
二重に例外を発生させないようにしよう	359
メソッドチェーンを使ってコードを書く	387
IDE (RubyMine) を使ってデバッグする	406
requireの単位はライブラリ	415
ワンライナーでRubyプログラムを実行する	419
Rubyやgemのバージョンとセキュリティ	421
ツールを使ったコードレビューの自動化	421
RubyとDSLの相性のよさ	424