

„Tato kniha, mimo jiné, skvěle vysvětluje metaprogramování, jeden z nejzajímavějších aspektů Ruby. Prvním vydáním této knihy bylo inspirováno mnoho raných myšlenek k Rails. Díky této knize se dostáváte na horskou dráhu mezi „Jak to můžu použít“ a „To je tak skvělé!“. Jakmile na tuto dráhu jednou nastoupíte, není už žádné cesty zpět.“

David Heinemeier Hansson,
Tvůrce Ruby on Rails,
Společník 37signals

„Druhé vydání této knihy je vzrušující událostí pro všechny programátory v Ruby – a pro milovníky skvělé technické literatury obecně. Hal Fulton přináší poutavý a jasný styl s důrazem na dokonalé a úzkostlivě přesné vysvětlení Ruby. Zřetelně cítíte přítomnost učitele, který má obrovské množství znalostí, a jenž vám chce opravdu pomoci, abyste je měli také.“

David Alan Black,
Autor Ruby for Rails

„Je to vynikající zdroj informací o tom, jak Ruby funguje. I já, jako člověk, který s Ruby pracuje po několik let, jsem zde našel plno nových triků a technik. Tuto knihu je možné nejenom číst od začátku do konce, ale také ji používat jako referenční příručku, ke které můžete kdykoliv sáhnout a naučit se něco nového.“

Chet Hendrickson,
Průkopník agilního softwaru

„Často používám první vydání této knihy o Ruby, abych zjistil podrobnosti o tomto programovacím jazyku, protože pokrývá mnoho témat, která nemohu najít nikde jinde. Nové vydání je ještě komplexnější a bude ještě hodnotnější.“

Ron Jeffries,
Autor a řečník

„Ruby je báječný jazyk, ale někdy prostě potřebujete jen něco udělat. Halova kniha vám poskytne řešení a poučí vás, proč je tohle řešení správné Ruby.“

Martin Fowler,
Chief Scientist, ThoughtWorks,
Autor „Patterns of Enterprise
Application Architecture“

Ruby

**kompedium znalostí pro
začátečníky i profesionály**

Hal Fulton

◆◆ Addison-Wesley



THE RUBY WAY, SECOND EDITION

Hal Fulton

Authorized translation from English language edition, entitled RUBY WAY, SECOND EDITION, THE: SOLUTION AND TECHNIQUES IN RUBY PROGRAMMING, 2nd Edition, 0672328844 by FULTON, HAL, published by Pearson Education, Inc, publishing as Addison Wesley Professional, Copyright © 2007. All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, INC. CZECH language edition by ZONER SOFTWARE S.R.O., Copyright © 2009.

Autorizovaný překlad anglického vydání nazvaného RUBY WAY, SECOND EDITION, THE: SOLUTION AND TECHNIQUES IN RUBY PROGRAMMING, druhé vydání, 0672328844, autor FULTON, HAL, vydal Pearson Education, Inc, ve vydavatelství Addison Wesley Professional, Copyright © 2007. Všechna práva vyhrazena. Žádná část této publikace nesmí být reprodukována nebo předávána žádnou formou nebo způsobem, elektronicky ani mechanicky, včetně fotokopíí, natáčení ani žádnými jinými systémy pro ukládání bez výslovného svolení Pearson Education, INC. České vydání ZONER SOFTWARE S.R.O., Copyright © 2009.

Ruby – kompendium znalostí pro začátečníky i profesionály

Autor: Hal Fulton

Copyright © ZONER software, s.r.o. Vydání první v roce 2009. Všechna práva vyhrazena.

Zoner Press

Katalogové číslo: **ZR724**

ZONER software, s.r.o.

Nové sady 18, 602 00 Brno

Překlad: Jiří Koutný

Odpovědný redaktor: Miroslav Kučera

Odborná korektura: Miroslav Kučera a Dalibor Šrámek

Šéfredaktor: Ing. Pavel Kristián

DTP: Miroslav Kučera

Zdrojové soubory ke knize:

<http://www.zonerpress.cz/download/ruby-kompendium.zip>

Informace, které jsou v této knize zveřejněny, mohou být chráněny jako patent. Jména produktů byla uvedena bez záruky jejich volného použití. Při tvorbě textů a vyobrazení bylo sice postupováno s maximální péčí, ale přesto nelze zcela vyloučit možnost výskytu chyb. Vydavatelé a autoři nepřebírají právní odpovědnost ani žádnou jinou záruku za použití chybných údajů a z toho vyplývajících důsledků. Všechna práva vyhrazena. Žádná část této publikace nesmí být reprodukována ani distribuována žádným způsobem ani prostředkem, ani reprodukována v databázi či na jiném záznamovém prostředku či v jiném systému bez výslovného svolení vydavatele, s výjimkou zveřejnění krátkých částí textu pro potřeby recenzí.

Veškeré dotazy týkající se distribuce směřujte na:

Zoner Press

ZONER software, s.r.o.

Nové sady 18, 602 00 Brno

tel.: **532 190 883**, fax: **543 257 245**

e-mail: **knihy@zoner.cz**

<http://www.zonerpress.cz>

ISBN 978-80-7413-018-2

Mým rodičům, bez kterých bych tu nebyl

Obsah

Předmluva	23
Předmluva k druhému vydání	23
Předmluva k prvnímu vydání	23
Poděkování	25
Poděkování pro druhé vydání	25
Poděkování pro první vydání	26
O autorovi	27
Úvod	28
O druhém vydání	28
Jak pracovat s touto knihou	31
Zdrojové kódy ke stažení	32
Poznámka redakce k českému vydání	33
Sdělte nám svůj názor	33
Co je "cesta Ruby"?	34
Kapitola 1	39
Ruby ve zkratce	
1.1 – Úvod do objektové orientace	40
1.1.1 – Co je objekt?	40
1.1.2 – Dědičnost	41
1.1.3 – Polymorfismus	43
1.1.4 – Ještě několik pojmů	44
1.2 – Základní syntaxe a sémantika Ruby	45
1.2.1 – Klíčová slova a identifikátory	46
1.2.2 – Komentáře a vestavěná dokumentace	46
1.2.3 – Konstanty, proměnné a typy	47
1.2.4 – Operátory a priorita	49
1.2.5 – Ukázkový program	50
1.2.6 – Cykly a větvení	53
1.2.7 – Výjimky	57
1.3 – OOP v Ruby	59
1.3.1 – Objekty	60
1.3.2 – Zabudované třídy	60
1.3.3 – Moduly a mixiny	62
1.3.4 – Vytváření tříd	62

1.3.5 – Metody a atributy	67
1.4 – Dynamické aspekty Ruby	68
1.4.1 – Programování v době běhu programu	69
1.4.2 – Reflexe	70
1.4.3 – Chybějící metody	72
1.4.4 – Svoz odpadků (garbage collection)	72
1.5 – Trénink intuice: co si zapamatovat	73
1.5.1 – Záležitosti syntaxe	73
1.5.2 – Perspektiva při programování	75
1.5.3 – Příkaz case v Ruby	78
1.5.4 – Rubyismy a idiomy	81
1.5.5 – Výrazy a další rozmanité záležitosti	86
1.6 – Hantýrka Ruby a slang	88
1.7 – Závěr	91

Kapitola 2	Práce s řetězci	93
-------------------	------------------------	-----------

2.1 – Reprezentace běžných řetězců	94
2.2 – Reprezentace řetězců s alternativní notací	94
2.3 – Víceřádkové řetězce (here-documents)	95
2.4 – Zjištění délky řetězce	97
2.5 – Zpracování řetězce po řádcích	97
2.6 – Zpracování řetězce po bajtech	97
2.7 – Provádění specializovaného porovnávání řetězců	98
2.8 – Rozdělení řetězce na znaky	100
2.9 – Formátování řetězce	101
2.10 – Řetězce jako IO objekty	102
2.11 – Malá a velká písmena	102
2.12 – Zpřístupňování a přiřazování podřetězců	103
2.13 – Nahrazování v řetězcích	105
2.14 – Prohledávání řetězce	106
2.15 – Konverze mezi znaky a ASCII kódy	107
2.16 – Implicitní a explicitní konverze	108
2.17 – Připojení položky do řetězce	110
2.18 – Odstranění přebytečných znaků pro nový řádek a dalších znaků	110
2.19 – Odstranění prázdných znaků z řetězce	111
2.20 – Opakování řetězců	112

2.21 – Vkládání výrazů do řetězců	112
2.22 – Odložené vyhodnocení při vkládání výrazu	113
2.23 – Analýza dat oddělených čárkou	113
2.24 – Konverze řetězců na čísla (dekadická a jiná)	114
2.25 – Kódování a dekódování textu rot13	116
2.26 – Šifrování řetězců	117
2.27 – Komprimace řetězců	117
2.28 – Počítání znaků v řetězci	118
2.29 – Obrácení řetězce	119
2.30 – Odstraňování duplicitních znaků	119
2.31 – Odstraňování znaků	119
2.32 – Tisk speciálních znaků	120
2.33 – Generování po sobě jdoucích řetězců	120
2.34 – Výpočet 32bitového CRC	121
2.35 – Výpočet haše řetězce pomocí MD5	121
2.36 – Výpočet Levenshteinovy vzdálenosti mezi dvěma řetězci	122
2.37 – Kódování a dekódování řetězců base64	124
2.38 – Kódování a dekódování řetězce (uuencode/uudecode)	125
2.39 – Konverze znaků Tab na mezery a naopak	125
2.40 – Zalomení řádků textu	126
2.41 – Závěr	127

Kapitola 3	Práce s regulárními výrazy	129
3.1 – Syntaxe regulárních výrazů		129
3.2 – Kompilování regulárních výrazů		131
3.3 – Ošetření speciálních znaků		132
3.4 – Používání kotev (anchors)		133
3.5 – Používání kvantifikátorů		134
3.6 – Pozitivní a negativní vyhlížení		136
3.7 – Přístup ke zpětným referencím		137
3.8 – Používání tříd znaků		140
3.9 – Rozšířené regulární výrazy		141
3.10 – Zachytávání znaku pro nový řádek pomocí tečky		142
3.11 – Použití modifikátoru na část regulárního výrazu		143
3.12 – Použití vložených podvýrazů		143
3.13 – Ruby a Oniguruma		144

3.13.1 – Testování přítomnosti Onigurumy	144
3.13.2 – Přidání Onigurumy	145
3.13.3 – Několik nových funkcionalit enginu Oniguruma	146
3.13.4 – Pozitivní a negativní zpětné vyhlížení	146
3.13.5 – Více o kvantifikátorech	147
3.13.6 – Pojmenované výsledky porovnání	148
3.13.7 – Rekurse v regulárních výrazech	149
3.14 – Několik ukázkových regulárních výrazů	150
3.14.1 – Porovnání IP adresy	150
3.14.2 – Porovnání páru atribut=hodnota	151
3.14.3 – Porovnání římských číslic	152
3.14.4 – Porovnání číselných konstant	153
3.14.5 – Porovnání řetězce datum/čas	153
3.14.6 – Detekce duplicitních slov v textu	154
3.14.7 – Porovnání slov psaných velkými písmeny	154
3.14.8 – Porovnání čísel verzí	155
3.14.9 – Několik dalších vzorů	155
3.15 – Závěr	156

Kapitola 4	Internacionalizace v Ruby	157
-------------------	----------------------------------	------------

4.1 – Pozadí a terminologie	158
4.2 – Kódování ve světě po éře ASCII	161
4.2.1 – Knihovna jcode a \$KCODE	162
4.2.2 – Běžné operace s řetězcí a regulárními výrazy	163
4.2.3 – Detekce kódování znaků	167
4.2.4 – Normalizace řetězců Unicode	167
4.2.5 – Problémy s řazením řetězců	169
4.2.6 – Konverze mezi kódováními	172
4.3 – Používání katalogů zpráv	174
4.3.1 – Pozadí a terminologie	175
4.3.2 – Začínáme s katalogy zpráv	175
4.3.3 – Lokalizace jednoduché aplikace	176
4.3.4 – Další poznámky	180
4.4 – Závěr	181

Kapitola 5	Vykonávání číselných výpočtů	183
5.1 – Reprezentace čísel v Ruby		183
5.2 – Základní operace s čísly		184
5.3 – Zaokrouhlování hodnot s pohyblivou řádovou čárkou		185
5.4 – Porovnání čísel s pohyblivou řádovou čárkou		187
5.5 – Formátování hodnot pro výstup		189
5.6 – Formátování čísel s čárkou		189
5.7 – Práce s velmi velkými celými čísly		190
5.8 – Použití BigDecimal		190
5.9 – Práce s racionálními čísly		192
5.10 – Práce s maticemi		193
5.11 – Práce s komplexními čísly		197
5.12 – Používání knihovny mathn		198
5.13 – Hledání faktorizace, GCD a LCM		199
5.14 – Práce s prvočíslly		200
5.15 – Implicitní a explicitní číselné konverze		201
5.16 – Vynucení číselné hodnoty		202
5.17 – Provádění bitových operací na číslech		203
5.18 – Provádění konverze základů		205
5.19 – Hledání odmocnin		205
5.20 – Určení pořadí bajtů architektury		206
5.21 – Numerický výpočet určitého integrálu		207
5.22 – Trigonometrie ve stupních, radiánech a gradiánech		208
5.23 – Pokročilejší trigonometrie		209
5.24 – Výpočet logaritmu s libovolným základem		209
5.25 – Výpočet průměru, mediánu, a modusu ze souboru dat		210
5.26 – Rozptyl a směrodatná odchylka		211
5.27 – Výpočet korelačního koeficientu		212
5.28 – Generování náhodných čísel		213
5.29 – Knihovna memoize pro cachování		214
5.30 – Závěr		215
Kapitola 6	Symbody a rozsahy	217
6.1 – Symbody		217
6.1.1 – Symbody jako výčty		219
6.1.2 – Symbody jako metahodnoty		219

6.1.3 – Symboly, proměnné a metody	220
6.1.4 – Konverze na/z symboly	221
6.2 – Rozsahy	222
6.2.1 – Otevřené a uzavřené rozsahy	222
6.2.2 – Hledání koncových bodů	223
6.2.3 – Iterace přes rozsah	223
6.2.4 – Testování příslušnosti k rozsahu	224
6.2.5 – Konverze na pole	224
6.2.6 – Opačné rozsahy	225
6.2.7 – Flip-flop operátor	225
6.2.8 – Uživatelské rozsahy	228
6.3 – Závěr	231

Kapitola 7	Práce s datem a časem	233
7.1 – Určení aktuálního času		234
7.2 – Práce se specifickými časy		234
7.3 – Určení dne v týdnu		235
7.4 – Určení data Velikonoc		236
7.5 – Hledání n-tého dne v měsíci		237
7.6 – Konverze mezi sekundami a většími jednotkami		238
7.7 – Konverze na a z epochy		238
7.8 – Práce s přestupnými sekundami? Raději ne!		239
7.9 – Nalezení čísla dne v roce		239
7.10 – Ověřování data a času		240
7.11 – Hledání týdne v roce		240
7.12 – Detekce přestupných roků		241
7.13 – Získávání časových pásem		242
7.14 – Práce s hodinami a minutami		242
7.15 – Porovnávání datových a časových hodnot		243
7.16 – Přidávání intervalů k datovým a časovým hodnotám		243
7.17 – Výpočet rozdílu dvou datových/časových hodnot		244
7.18 – Práce se specifickými daty		244
7.19 – Konverze mezi třídami Time, Date a DateTime		245
7.20 – Získávání hodnot data a času z řetězce		246
7.21 – Formátování a tisk hodnot data a času		247
7.22 – Konverze časových pásem		248

7.23 – Určení počtu dnů v měsíci	248
7.24 – Rozdělení měsíce na jednotlivé týdny	249
7.25 – Závěr	250

Kapitola 8	Pole, haš a ostatní výčty	251
-------------------	----------------------------------	------------

8.1 – Práce s poli	251
8.1.1 – Vytvoření a inicializace pole	252
8.1.2 – Zpřístupnění a přiřazení prvků pole	252
8.1.3 – Nalezení velikosti pole	254
8.1.4 – Porovnávání polí	254
8.1.5 – Řazení polí	256
8.1.6 – Výběr z pole na základě kritéria	259
8.1.7 – Použití specializovaných indexovacích funkcí	260
8.1.8 – Implementace řídkých matic	262
8.1.9 – Pole jako matematické množiny	263
8.1.10 – Náhodné uspořádání prvků pole	266
8.1.11 – Používání vícerozměrných polí	267
8.1.12 – Hledání prvků pole, které nejsou prvky jiného	268
8.1.13 – Transformace nebo mapování polí	269
8.1.14 – Odstranění hodnot nil z pole	269
8.1.15 – Odstranění specifických prvků z pole	269
8.1.16 – Zřetězení a připojení do polí	271
8.1.17 – Použití pole jako zásobníku nebo fronty	272
8.1.18 – Iterace skrz pole	272
8.1.19 – Oddělovače prvků pole při tvorbě řetězce	273
8.1.20 – Převrácení pole	273
8.1.21 – Odstranění duplicitních prvků z pole	274
8.1.22 – Prokládání polí	274
8.1.23 – Zjišťování počtu výskytů hodnot v poli	274
8.1.24 – Převrácení pole do podoby haše	275
8.1.25 – Synchronizované řazení několika polí	275
8.1.26 – Nastavení výchozí hodnoty pro nové prvky pole	276
8.2 – Práce s hašem	277
8.2.1 – Vytvoření nového haše	277
8.2.2 – Nastavení výchozí hodnoty pro haš	278
8.2.3 – Zpřístupnění a přidávání dvojic klíč-hodnota	279

8.2.4 – Odstranění dvojic klíč-hodnota	280
8.2.5 – Iterace skrz haš	281
8.2.6 – Invertování haše	281
8.2.7 – Detekce klíčů a hodnot v haši	281
8.2.8 – Extrahování haše do polí	282
8.2.9 – Výběr dvojice klíč-hodnota na základě kritéria	282
8.2.10 – Řazení haše	283
8.2.11 – Slučování dvou hašů	283
8.2.12 – Vytvoření haše z pole	284
8.2.13 – Hledání rozdílu nebo průniku klíčů haše	284
8.2.14 – Použití haše jako řídké matice	284
8.2.15 – Implementace haše s duplicitními klíči	285
8.3 – Výčty obecně	288
8.3.1 – Metoda inject	289
8.3.2 – Používání kvantifikátorů	290
8.3.3 – Metoda partition	290
8.3.4 – Iterování přes skupiny	291
8.3.5 – Konverze na pole nebo množiny	292
8.3.6 – Používání enumerátorů	293
8.3.7 – Používání generátorů	294
8.4 – Závěr	295

Kapitola 9 Pokročilejší datové struktury 297

9.1 – Práce s množinami	297
9.1.1 – Jednoduché množinové operace	298
9.1.2 – Pokročilejší množinové operace	299
9.2 – Práce se zásobníky a frontami	301
9.2.1 – Implementace přísnějšího zásobníku	302
9.2.2 – Detekce nevyrovnaného použití párových znaků	303
9.2.3 – Zásobníky a rekurze	304
9.2.4 – Implementace přísnější fronty	305
9.3 – Práce se stromy	306
9.3.1 – Implementace binárního stromu	307
9.3.2 – Řazení s použitím binárního stromu	309
9.3.3 – Používání binárního stromu jako vyhledávací tabulky	311
9.3.4 – Konverze stromu na řetězec nebo pole	311

9.4 – Práce s grafy	312
9.4.1 – Implementace grafu jako matice sousednosti	313
9.4.2 – Určení, zdali je graf souvislý	316
9.4.3 – Určení, zdali má graf Eulerovu kružnici	317
9.4.4 – Určení, zdali má graf Eulerovu cestu	318
9.4.5 – Nástroje Ruby pro práci s grafy	319
9.5 – Závěr	319

Kapitola 10	I/O a uložení dat	321
--------------------	--------------------------	------------

10.1 – Práce se soubory a adresáři	322
10.1.1 – Otevírání a zavírání souborů	322
10.1.2 – Aktualizace souboru	323
10.1.3 – Připojení k souboru	324
10.1.4 – Náhodný přístup k souborům	324
10.1.5 – Práce s binárními soubory	325
10.1.6 – Zamykání souborů	327
10.1.7 – Vykonávání jednoduchých I/O	327
10.1.8 – Bufferované a nebufferované I/O	328
10.1.9 – Manipulace s vlastnictvím a přístupovými právy souboru	329
10.1.10 – Získávání a nastavování informací o časových údajích	331
10.1.11 – Kontrola existence souboru a velikosti	332
10.1.12 – Speciální vlastnosti souborů	333
10.1.13 – Práce s rourami (pipes)	335
10.1.14 – Vykonávání speciálních I/O operací	337
10.1.15 – Používání neblokujících I/O	337
10.1.16 – Použití metody readpartial	338
10.1.17 – Manipulace s cestami	339
10.1.18 – Používání třídy Pathname	340
10.1.19 – Manipulace se soubory na úrovni příkazů	341
10.1.20 – Čtení znaků z klávesnice	342
10.1.21 – Načtení celého souboru do paměti	343
10.1.22 – Iterace skrz soubor po řádcích	343
10.1.23 – Iterace skrz soubor po bajtech	344
10.1.24 – Práce s řetězcem jako se souborem	344
10.1.25 – Čtení dat vestavěných v programu	345
10.1.26 – Čtení zdroje programu	345

10.1.27 – Práce s dočasnými soubory	346
10.1.28 – Změna a nastavení aktuálního adresáře	346
10.1.29 – Změna aktuálního kořene	347
10.1.30 – Iterace skrz položky adresáře	347
10.1.31 – Získání seznamu položek adresáře	347
10.1.32 – Vytvoření sledu adresářů	348
10.1.33 – Rekurzivní odstranění adresáře	348
10.1.34 – Hledání souborů a adresářů	348
10.2 – Vysokourovňový přístup k datům	349
10.2.1 – Jednoduchý marshaling	349
10.2.2 – Složitější marshaling	351
10.2.3 – Omezené rekurzivní kopírování pomocí modulu Marshal	352
10.2.4 – Lepší perzistence objektů s PStore	352
10.2.5 – Práce s daty ve formátu CSV	354
10.2.6 – Marshaling s YAML	356
10.2.7 – Prevalence objektů s Madeleine	357
10.2.8 – Použití knihovny DBM	358
10.3 – Použití knihovny KirbyBase	359
10.4 – Spojení s externími databázemi	362
10.4.1 – Rozhraní k SQLite	362
10.4.2 – Rozhraní k MySQL	363
10.4.3 – Rozhraní k PostgreSQL	366
10.4.4 – Rozhraní k LDAP	368
10.4.5 – Rozhraní k Oracle	370
10.4.6 – Používání modulu DBI	371
10.4.7 – ORM (Object-Relational Mapper)	372
10.5 – Závěr	374

Kapitola 11	OOP a dynamické rysy Ruby	375
--------------------	----------------------------------	------------

11.1 – Každodenní OOP úlohy	376
11.1.1 – Používání vícenásobných konstruktorů	376
11.1.2 – Vytváření atributů instance	377
11.1.3 – Používání pokročilých konstruktorů	378
11.1.4 – Vytváření vlastností a metod na úrovni třídy	380
11.1.5 – Dědění z nadtřídy	383
11.1.6 – Testování třídy objektů	385

11.1.7 – Testování rovnosti objektů	387
11.1.8 – Správa přístupu k metodám	389
11.1.9 – Kopírování objektu	391
11.1.10 – Používání initialize_copy	392
11.1.11 – Jak porozumět metodě allocate	393
11.1.12 – Práce s moduly	394
11.1.13 – Transformace a konverze objektů	397
11.1.14 – Vytváření čistě datových tříd (Structs)	400
11.1.15 – Zmrazení objektů	401
11.2 – Pokročilejší techniky	402
11.2.1 – Posílání explicitních zpráv objektu	403
11.2.2 – Specializování individuálních objektů	404
11.2.3 – Vnořování tříd a modulů	408
11.2.4 – Vytváření parametrických tříd	408
11.2.5 – Využití pokračování (continuation) pro implementaci generátoru	411
11.2.6 – Ukládání kódu jako objektů	414
11.2.7 – Jak funguje přidání modulu	415
11.2.8 – Detekce výchozích parametrů	417
11.2.9 – Delegování nebo přesměrování	417
11.2.10 – Automatické definování přístupových metod k atributům na úrovni třídy	420
11.2.11 – Pokročilejší programovací disciplíny	421
11.3 – Práce s dynamickými rysy	423
11.3.1 – Dynamické vyhodnocení kódu	423
11.3.2 – Používání const_get	425
11.3.3 – Dynamická instanciací třídy podle jména	425
11.3.4 – Získávání a nastavování proměnných instance	426
11.3.5 – Používání define_method	427
11.3.6 – Používání const_missing	430
11.3.7 – Odstraňování definic	431
11.3.8 – Získávání seznamů definovaných entit	434
11.3.9 – Prozkoumání zásobníku	435
11.3.10 – Monitorování vykonávání programu	436
11.3.11 – Procházení prostorem objektů	438
11.3.12 – Správa volání neexistujících metod	438
11.3.13 – Sledování změn definice třídy nebo objektů	439
11.3.14 – Definování finalizerů pro objekty	443
11.4 – Závěr	444

Kapitola 12 Grafická rozhraní pro Ruby

445

12.1 – Ruby/Tk	446
12.1.1 – Přehled	446
12.1.2 – Jednoduchá aplikace s okny	447
12.1.3 – Práce s tlačítky	449
12.1.4 – Práce s textovými poli	452
12.1.5 – Práce s ostatními widgety	456
12.1.6 – Několik poznámek	459
12.2 – Ruby/GTK2	459
12.2.1 – Přehled	460
12.2.2 – Jednoduchá aplikace s okny	461
12.2.3 – Práce s tlačítky	462
12.2.4 – Práce s textovými poli	463
12.2.5 – Práce s ostatními widgety	466
12.2.6 – Další poznámky	470
12.3 – FXRuby (FOX)	472
12.3.1 – Přehled	473
12.3.2 – Jednoduchá aplikace s okny	473
12.3.3 – Práce s tlačítky	475
12.3.4 – Práce s textovými poli	477
12.3.5 – Práce s dalšími widgety	478
12.3.6 – Další poznámky	487
12.4 – QtRuby	487
12.4.1 – Přehled	487
12.4.2 – Jednoduchá aplikace s okny	488
12.4.3 – Práce s tlačítky	488
12.4.4 – Práce s textovými poli	490
12.4.5 – Práce s ostatními widgety	492
12.4.6 – Další poznámky	497
12.5 – Další GUI toolkity	497
12.5.1 – Ruby a X	498
12.5.2 – Ruby a wxWidgets	498
12.5.3 – Apollo (Ruby a Delphi)	498
12.5.4 – Ruby a Windows API	499
12.6 – Závěr	499

Kapitola 13	Vlákna v Ruby	501
13.1 – Vytváření a manipulace s vlákny		502
13.1.1 – Vytvoření vláken		502
13.1.2 – Přístup k lokálním proměnným vlákna		503
13.1.3 – Dotazování a změna stavu vlákna		505
13.1.4 – Čekání na dokončení (a zachycení návratové hodnoty)		508
13.1.5 – Práce s výjimkami		509
13.1.6 – Použití skupiny vláken		511
13.2 – Synchronizace vláken		512
13.2.1 – Jednoduchá synchronizace s kritickými sekcemi		513
13.2.2 – Synchronizace přístupu ke zdrojům (mutex.rb)		514
13.2.3 – Použití tříd předdefinovaných synchronizovaných front		518
13.2.2 – Použití podmínkových proměnných		520
13.2.5 – Použití dalších synchronizačních technik		521
13.2.6 – Časový limit operace		524
13.2.7 – Čekání na událost		525
13.2.8 – Pokračování ve zpracování během I/O		526
13.2.9 – Implementace paralelních iterátorů		527
13.2.10 – Paralelní rekurzivní mazání		529
13.3 – Závěr		530
Kapitola 14	Skriptování a správa systému	531
14.1 – Spuštění externího programu		531
14.1.1 – Použití system a exec		532
14.1.2 – Substituce výstupu příkazu		533
14.1.3 – Manipulace s procesy		534
14.1.4 – Manipulace se standardním vstupem/výstupem		536
14.2 – Volby a argumenty příkazového řádku		537
14.2.1 – Analýza voleb příkazového řádku		537
14.2.2 – Práce s ARGF		539
14.2.3 – Práce s ARGV		539
14.3 – Knihovna Shell		540
14.3.1 – Použití Shell pro přesměrování I/O		540
14.3.2 – Několik poznámek k shell.rb		542
14.4 – Přístup k proměnným prostředí		543
14.4.1 – Získávání a nastavování proměnných prostředí		543

14.4.2 – Ukládání proměnných prostředí jako pole nebo haš	544
14.4.3 – Import proměnných prostředí jako globálních proměnných	545
14.5 – Skriptování v Microsoft Windows	545
14.5.1 – Používání Win32API	546
14.5.2 – Použití Win32OLE	547
14.5.3 – Používání ActiveScriptRuby	550
14.6 – Instalátor na jedno kliknutí pro Windows	551
14.7 – Knihovny, o kterých potřebujete vědět	552
14.8 – Práce se soubory, adresáři a stromy	553
14.8.1 – Několik slov k textovým filtrům	553
14.8.2 – Kopírování adresářového hostromu (včetně symbolických odkazů)	554
14.8.3 – Mazání souborů podle data nebo jiného kritéria	555
14.8.4 – Zjištění volného místa na disku	556
14.9 – Různé skriptovací úlohy	557
14.9.1 – Ruby řešení v jediném souboru	557
14.9.2 – Roura do interpretu Ruby	558
14.9.3 – Získávání a nastavování návratových kódů	559
14.9.4 – Testování, jestli program běží interaktivně	560
14.9.5 – Určení platformy operačního systému	560
14.9.6 – Použití modulu Etc	561
14.10 – Závěr	562

Kapitola 15	Ruby a datové formáty	563
--------------------	------------------------------	------------

15.1 – Analýza XML pomocí REXML	564
15.1.1 – Analýza stromu	565
15.1.2 – Analýza proudu	566
15.1.3 – XPath a další	567
15.2 – Práce s RSS a Atom	568
15.2.1 – Standardní knihovna rss	568
15.2.2 – Knihovna feedtools	571
15.3 – Manipulace s obrázky pomocí RMagick	572
15.3.1 – Běžné grafické úkoly	573
15.3.2 – Speciální efekty a transformace	576
15.3.3 – Kreslicí API	579
15.4 – Tvorba PDF dokumentů pomocí PDF::Writer	582
15.4.1 – Základní koncepty a techniky	582

15.4.2 – Vzorový dokument	585
15.5 – Závěr	592
Kapitola 16	Testování a odstraňování chyb
593	
16.1 – Testování pomocí Test::Unit	594
16.2 – Nástroje ZenTest	598
16.3 – Ruby debugger	601
16.4 – Používání irb jako debuggeru	605
16.5 – Měření vytížení kódu	606
16.6 – Měření výkonu	607
16.7 – Uživatelsky příjemná reprezentace objektů	611
16.8 – Závěr	612
Kapitola 17	Balíčkování a distribuce kódu
613	
17.1 – Používání RDoc	613
17.1.1 – Jednoduché značky	615
17.1.2 – Pokročilejší formátování	617
17.2 – Instalace a balíčkování	618
17.2.1 – Knihovna setup.rb	619
17.2.2 – RubyGems	620
17.3 – RubyForge a RAA	622
17.4 – Závěr	623
Kapitola 18	Síťové programování
625	
18.1 – Síťové servery	626
18.1.1 – Jednoduchý server: aktuální čas	627
18.1.2 – Implementace serveru s vlákny	628
18.1.3 – Případová studie: peer-to-peer šachový server	629
18.2 – Síťoví klienti	638
18.2.1 – Získávání opravdu náhodných čísel z webu	638
18.2.2 – Spojení s oficiálním časovým serverem	641
18.2.3 – Interakce s POP serverem	642
18.2.4 – Posílání e-mailu prostřednictvím SMTP	644
18.2.5 – Interakce s IMAP serverem	647
18.2.6 – Kódování a dekodování příloh	648
18.2.7 – Případová studie: brána mezi e-mailovou konferencí a diskusní skupinou	651

18.2.8 – Získávání webové stránky z URL	656
18.2.9 – Používání knihovny Open-URI	657
18.3 – Závěr	657

Kapitola 19	Ruby a webové aplikace	659
--------------------	-------------------------------	------------

19.1 – CGI programování v Ruby	659
19.1.1 – Představení knihovny cgi.rb	661
19.1.2 – Zobrazení a zpracování formulářů	662
19.1.3 – Práce s cookies	663
19.1.4 – Práce s relacemi uživatele	664
19.2 – Používání FastCGI	665
19.3 – Ruby on Rails	667
19.3.1 – Principy a technologie	667
19.3.2 – Testování a ladění Rails aplikací	669
19.3.3 – Rozšíření jádra	669
19.3.4 – Související nástroje a knihovny	670
19.4 – Vývoj webových aplikací s Nitro	671
19.4.1 – Vytvoření základní Nitro aplikace	671
19.4.2 – Nitro a vzor MVC	673
19.4.3 – Nitro a Og	677
19.4.4 – Běžné úkoly při vývoji webových aplikací v Nitro	678
19.4.5 – Několik důležitých detailů	681
19.5 – Úvod do Wee	683
19.5.1 – Jednoduchý příklad	684
19.5.2 – Asociování stavu s URL	685
19.6 – Vývoj webových aplikací s IOWA	686
19.6.1 – Základní koncepty IOWA	686
19.6.2 – Šablony v IOWA	689
19.6.3 – Předávání řízení mezi komponentami	690
19.7 – Ruby a webový server	691
19.7.1 – Používání mod_ruby	692
19.7.2 – Používání erb	693
19.7.3 – Používání serveru WEBrick	695
19.7.4 – Používání serveru Mongrel	697
19.8 – Závěr	699

Kapitola 20	Distribuované Ruby	701
20.1 – Přehled: použití drb		702
20.2 – Případová studie: simulace burzovního telegrafu		704
20.3 – Rinda: Ruby Tuplespace		708
20.4 – Service Discovery s distribuovaným Ruby		712
20.5 – Závěr		713
Kapitola 21	Vývojové nástroje pro Ruby	715
21.1 – Použití RubyGems		715
21.2 – Použití Rake		717
21.3 – Použití irb		721
21.4 – Utilita ri		726
21.5 – Podpora editorů		727
21.6 – Integrovaná vývojová prostředí		728
21.7 – Závěr		729
Kapitola 22	Komunita Ruby	731
22.1 – Zdroje na webu		731
22.2 – Diskusní skupiny a e-mailové konference		732
22.3 – Blogy a online magazíny		732
22.4 – Ruby Change Requests (požadavky na změnu)		733
22.5 – IRC kanály		733
22.6 – Konference o Ruby		734
22.8 – Závěr		734
Rejstřík		735

Předmluva

Předmluva k druhému vydání

Lidé – zejména filozofové – ve starověké Číně si mysleli, že za světem a za každou existencí je ukryto něco tajemného. Něco, co nemůže být nikdy vyřčeno, vysvětleno, a ani popsáno slovy. Číňané to nazývali Tao, Japonci zase Do. Pokud to přeložíte do angličtiny, znamená to Way (v češtině "Cesta" nebo "Způsob"). Je to Do v džudo, kendo, karatedo a aikido. Nejsou to jen bojová umění, ale zahrnují jak filozofii, tak i způsob života.

Také programovací jazyk Ruby má svou filozofii a způsob myšlení. Naučí lidi myslet jinak. Pomáhá programátorům, aby měli při své práci více zábavy. Není to tím, že Ruby pochází z Japonska, ale tím, že programování je důležitou částí lidského bytí (dobře, přinejmenším některých lidských bytostí), přičemž Ruby bylo navrženo k tomu, aby lidem pomáhalo mít lepší život.

Tao je těžké popsat. Ačkoliv ho cítím, nikdy jsem se nepokusil jej vysvětlit pomocí slov. Je to pro mě příliš obtížné, dokonce i v japonštině, v mém rodném jazyce. Ale chlapík jménem Hal Fulton to zkusil a jeho první pokus, který spočíval v prvním vydání této knihy, byl dost dobrý. Tento jeho druhý pokus popsat Tao jazyka Ruby je díky pomoci mnoha lidí z komunity ještě lepší. Jak se Ruby stává stále více populárním (částečně i kvůli Ruby on Rails), je důležité porozumět tajemství programátorské produktivity. Věřím, že tato kniha vám pomůže se stát zdatným programátorem.

Veselé hackování.

Yukihiro "Matz" Matsumoto

Srpen 2006, Japonsko

まつもと ゆきひろ

Předmluva k prvnímu vydání

Krátce poté, co jsem se počátkem 80. let poprvé setkal s počítači, začal jsem se zajímat o programovací jazyky. Od té doby jsem na ně byl zatížený. Myslím si, že důvodem pro tento zájem je skutečnost, že programovací jazyky jsou způsobem, jak vyjádřit lidské myšlenky. Jsou v zásadě orientovány na člověka.

Navzdory tomuto faktu měly programovací jazyky sklon být orientovány strojově. Mnoho jazyků bylo navrženo pro pohodlí samotných počítačů.

Ale vzhledem k tomu, že se počítače staly výkonnějšími a méně nákladnými, se tato situace postupně změnila. Podívejme se například na strukturované programování. Stroje se nestarají o to, zdali je program strukturován správně – pouze ho vykonají bit po bitu. Strukturované programování není pro stroje, ale pro lidi. Totéž platí pro objektově orientované programování.

Přišel čas pro návrh jazyka zaměřeného na člověka.

V roce 1993 jsem mluvil s kolegou o skriptovacích jazycích, o jejich síle a budoucnosti. Skriptování jsem cítil jako cestu, kterou by se programování mělo v budoucnosti ubírat – orientací na člověka.

Ale s existujícími jazyky jako Perl a Python jsem nebyl spokojen. Chtěl jsem jazyk, který by byl silnější než Perl a více objektově orientovaný než Python. Protože jsem nemohl najít ideální jazyk, rozhodl jsem se vytvořit svůj vlastní.

Ruby sice není nejjednodušším jazykem, ale lidská duše ve svém přirozeném stavu také není jednoduchá. Miluje jednoduchost a složitost zároveň. Nemůže ovládat příliš mnoho komplexních věcí, ale ani příliš mnoho věcí jednoduchých. Je to věc rovnováhy.

Při navrhování jazyka orientovaného na člověka, Ruby, jsem následoval princip nejmenšího překvapení (Principle of Least Surprise). Všechno, co mě překvapilo, jsem považoval za méně správné. Následkem toho mám při programování v Ruby dobrý pocit – dokonce druh radosti. A již od prvního vydání Ruby v roce 1995 se mnou mnoho programátorů po celém světě o radosti z programování v Ruby souhlasí.

Zde bych rád vyjádřil mé velké uznání lidem v komunitě Ruby. Oni jsou srdcem úspěchu Ruby.

Jsem také vděčný autorovi této knihy, kterým je Hal E. Fulton.

Tato kniha vysvětluje filozofii stojící za Ruby destilovanou z mého mozku a z komunity Ruby. Rád bych věděl, jak je možné, že Hal může číst mou mysl a odhalit tak tajemství Ruby. Nikdy jsem se s ním nesetkal tváří v tvář, ale doufám, že k tomu brzy dojde.

Věřím, že tato kniha a Ruby vám pomohou v tom, aby vaše programování bylo zábavou.

Yukihiro "Matz" Matsumoto

Září 2001, Japonsko

まつもと ゆきひろ

Poděkování

Poděkování pro druhé vydání

Zdravý rozum říká, že druhé vydání bude vyžadovat dvakrát méně práce, než vyžadovalo vydání první. Zdravý rozum nemá pravdu.

Třebaže velká část této knihy vychází z prvního vydání, musela být část přepracována a doladěna. Každá jednotlivá věta v této knize musela projít skrz filtr, který se ptal: "Co bylo pravdou v roce 2001, je pravdou i v roce 2006?" A to je samozřejmě jen začátek.

Stručně řečeno – na toto druhé vydání jsem vynaložil nespočet stovek hodin práce; téměř tolik, kolik jsem vynaložil na první vydání. A to jsem "pouze autor".

Tato kniha mohla vzniknout pouze díky kolektivní práci mnoha lidí. Na straně vydavatele dlužím poděkováním těmto lidem za jejich tvrdou práci a nekonečnou trpělivost: Debra Williams Cauley, Songlin Qui a Mandie Frank. Další díky patří Geneil Breezeové za její neúnavné úpravy mé poněkud nedokonalé angličtiny. Chci zde poděkovat i všem spolupracovníkům, se kterými jsem se nikdy nesetkal, protože jejich práce probíhala v pozadí.

Technická redakce byla v tomto složení: Shashank Date a Francis Hwang. Odvedli skvělou práci a já si toho cením. Chyby, které se nakonec dostaly do tisku, samozřejmě padají na mou hlavu.

Mé další poděkování patří lidem, kteří dodávali vysvětlivky, psali vzorový kód a odpovídali na mé četné otázky. Mezi ně patří samotný Matz (Yukihiro Matsumoto), Dave Thomas, Christian Neukirchen, Chad Fowler, Curt Hibbs, Daniel Berger, Armin Roehrl, Stefan Schmiedl, Jim Weirich, Ryan Davis, Jenny W., Jim, Freeze, Lyle Johnson, Martin DeMello, Matt Lawrence, Ron Jeffries, Tim Hunter, Chet Hendrickson, Nathaniel Talbott a Bil Kleb.

Zvláštní dík patří největším přispěvatelům. Andrew Johnson podstatně zvýšil mou znalost regulačních výrazů. Paul Battley poskytl významné příspěvky do kapitoly o internacionalizaci. Masao Mutoh vypomohl se stejnou kapitolou a dále přispěl materiály o GTK. Austin Ziegler mě zasvětil do tajemství vytváření PDF souborů. Caleb Tennis přidal materiály o Qt. Eric Hodel přispěl materiálem o Rinda a Ring. James Britt vypomohl s kapitolou o webovém vývoji.

Znovu musím poděkovat a pochválit Matze – nejenom za jeho pomoc, ale v první řadě za vytvoření Ruby. Domo arigato gozaimasu! ("Mockrát děkuji!" – pozn. red.)

Opět děkuji svým rodičům, kteří mě bez přestání povzbuzovali a těšili se na tuto knihu. Možná ještě z nich udělám programátory.

A ještě jednou musím poděkovat všem lidem z komunity Ruby za jejich neúnavnou práci, produktivitu a za udržování ducha komunity. Obzvláště děkuji čtenářům této knihy (v obou vydáních) a doufám, že ji shledáte poučnou, užitečnou a možná dokonce i zábavnou.

Poděkování pro první vydání

Napsání knihy je týmové úsilí. Jedná se o skutečnost, kterou jsem nemohl plně docenit až do té doby, dokud jsem sám nějakou knihu nenapsal. Doporučuji to zažít, ačkoliv je to ponižující. Potvrzuji tedy, že bez podpory mnoha lidí by tato kniha nemohla existovat.

Mé poděkování a ocenění musí nejprve směřovat k Matzovi (Yukihiro Matsumoto), který vytvořil jazyk Ruby. Domo arigato gozaimasu!

Další dík patří Conradu Scheikerovi za pomoc při vytváření celkové struktury knihy. Tento člověk mi navíc prokázal velkou službu, když mě v roce 1999 uvedl do jazyka Ruby.

Na vytváření obsahu této knihy se samozřejmě podílelo několik dalších lidí. Zde musím především zmínit Guye Hursta, který napsal nejenom značné části úvodních kapitol, ale také dva dodatky ke knize. Jeho pomoc musím označit za neocenitelnou.

Poděkování pochopitelně patří i ostatním přispěvatelům, které zde uvádím bez nějakého speciálního pořadí: Kevin Smith odvedl skvělou práci na sekci věnované GTK v kapitole 6, čímž mě zachránil před horečným osvojováním si potřebných znalostí. Patrick Logan ve stejné kapitole osvětlil tajemství GUI FOX. Nesmím zapomenout na Chada Fowlera, který se v kapitole 9 věnoval XML, a jenž také přispěl nějakým materiálem do CGI sekce.

Velký dík také patří těm, kteří asistovali při provádění korektur, a již hodnotili nebo přispěli v dalších rozmanitých oblastech: Don Muchow, Mike Stok, Miho Ogishima a další. Také děkuji Davidu Eppsteinovi, profesoru matematiky, za zodpovězení mých otázek o teorii grafů.

Jednou z významných věcí na Ruby je podpora komunity. V e-mailových konferencích a diskusních skupinách se našlo mnoho lidí, kteří zodpověděli mé otázky a poskytli mi potřebnou pomoc. Opět bez nějakého speciálního pořadí se jednalo o tyto lidi: Dave Thomas, Andy Hunt, Hee-Sob Park, Mike Wilson, Avi Bryant, Yasushi Shoji ("Yashi"), Shugo Maeda, Jim Weirich, "Arton" a Masaki Suketa. Pokud jsem na někoho zapomněl, omlouvám se.

A samozřejmě – tato kniha by nikdy nevyšla bez ohromné podpory vydavatele. Na produkci této knihy v zákulisí pracovalo mnoho lidí – v první řadě musím poděkovat Williamu Brownovi, který pracoval blízko mě a byl stálým zdrojem podpory, a Scottu Meyerovi, jenž se důkladně probíral všemi podklady a dával jednotlivé materiály dohromady. Ostatní bohužel nemohu ani jmenovat, protože jsem o nich nikdy neslyšel. Nicméně oni sami vědí, o koho jde.

Také musím poděkovat svým rodičům, kteří z povzdálí tento projekt sledovali, po celou dobu mě povzbuzovali a dokonce se kvůli mně obtěžovali naučit kus počítačové vědy.

Jeden můj přítel, který pracuje jako spisovatel, mi jednou řekl: "Když napíšeš knihu a nikdo si ji nepřečte, pak jsi ve skutečnosti žádnou knihu nenapsal." Takže chci poděkovat i svým čtenářům. Tato kniha je určena vám. Doufám, že pro vás má nějakou hodnotu.

O autorovi

Hal Fulton má dva tituly v počítačové vědě na University of Mississippi. Předtím, než se kvůli řadě kontraktů (zejména pro IBM) přestěhoval do Austinu v Texasu, čtyři roky vyučoval počítačovou vědu na univerzitní úrovni. Více než 15 let pracoval s různými variantami Unixu, včetně AIX, Solaris a Linux. K Ruby se úplně poprvé dostal v roce 1999. V roce 2001 začal pracovat na prvním vydání této knihy, která byla tehdy celkově druhou knihou o Ruby v anglickém jazyce. Účastnil se celkem šesti konferencí o Ruby, přičemž na čtyřech z nich měl svou vlastní prezentaci (včetně první European Ruby Conference, která se konala v Karlsruhe v Německu). V současné době pracuje pro Broadwing Communications v Austinu v Texasu, kde pracuje na velkém datovém skladišti a souvisejících telekomunikačních aplikacích. Každý den pracuje s kódem C++, Oracle a samozřejmě i s Ruby.

Hal Fulton je aktivním členem e-mailové konference (a IRC kanálu) zaměřené na Ruby. Má rozpracovaných několik projektů týkajících se Ruby. Je členem ACM a IEEE Computer Society. V osobním životě má rád hudbu, čtení, umění a fotografování. Je členem Mars Society a je vesmírný nadšenec, který by se rád dostal do vesmíru předtím, než zemře. Žije v Austinu v Texasu.

Úvod

Cesta, kterou lze pojmenovat, není tou pravou Cestou.

– Lao C', *Kniha o Tao a ctnosti (Tao-te-ťing)*

Originální anglický název této knihy je "The Ruby Way". Tento název si říká o vysvětlení. Mým cílem bylo napsat tuto knihu pokud možno ve shodě se samotnou filozofií Ruby. To bylo také cílem všech ostatních spolupracovníků. Ačkoliv s nimi sdílím ocenění za úspěch, odpovědnost za jakékoliv chyby, které se do této knihy dostaly, leží výhradně na mně.

Samozřejmě není v mých silách, abych vám precizním způsobem řekl úplně všechno o Ruby. To je v první řadě úkol pro samotného tvůrce jazyka, Matze, ale myslím si, že dokonce i on by měl potížit vyjádřit vše slovy. Stručně řečeno: "Ruby – kompendium znalostí pro začátečníky i profesionály" je pouze kniha, ale cesta Ruby (Ruby Way) je věcí tvůrce jazyka a komunity jako celku. Ačkoliv tohle lze jen velmi těžko popsat pomocí slov, v tomto úvodu se pokusím zachytit alespoň něco z toho nevyslovitelného o Ruby. Moudrý zájemce o Ruby to nicméně nebude brát až tak úplně vážně.

Vezměte, prosím, na vědomí, že toto je druhé vydání. Ačkoliv mnoho věcí zůstalo stejných, mnoho dalších věcí se změnilo. Většina tohoto úvodu je stejná jako v předchozím vydání, nicméně nahlédněte do následující části "O druhém vydání", ve které jsou shrnuty změny a nový materiál.

O druhém vydání

Všechno se mění a Ruby není výjimkou. Když v srpnu 2006 píšu tento úvod, první vydání této knihy je téměř pět let staré. A to je určitě vhodný čas na aktualizaci.

V tomto vydání je mnoho změn a mnoho nových materiálů. Stará kapitola 4 o jednoduchých datových úlohách je nyní rozdělena do šesti kapitol, dvě z nich (Symboly a rozsahy a Internacionalizace v Ruby) jsou úplně nové. Ve zbývajících čtyřech jsou přidány nové příklady a komentáře. Kapitola o regulárních výrazech byla rapidně rozšířena – nyní pokrývá nejenom běžné regulární výrazy, ale také novější engine Oniguruma.

Obsah kapitol 8 a 9 byl v prvním vydání této knihy původně obsažen v jediné kapitole. Ta byla rozdělena v okamžiku, kdy došlo k přidání dalších materiálů, protože se příliš rozrostla.

Současné kapitoly 18, 19 a 20 obdobným způsobem vyrostly z kapitoly 9. Abychom získali místo navíc pro tyto materiály, museli jsme bohužel z nového vydání odstranit všechny přílohy.

Zbývajících nových kapitol jsou následující:

- **Kapitola 15 – Ruby a datové formáty.** Zahrnuje XML, RSS, obrázkové soubory, tvorbu PDF souborů a další věci.
- **Kapitola 16 – Testování a odstraňování chyb.** Zabývá se unit testy, profilováním, laděním a dalšími podobnými tématy.

- **Kapitola 17 – Balíčkování a distribuce kódu.** Tato kapitola zahrnuje použití `setup.rb`, vytvoření RubyGems a další záležitosti.
- **Kapitola 21 – Vývojové nástroje pro Ruby.** Nabízí pohled na podporu Ruby v editoru a IDE, utilitu `ri` a RubyGems z pohledu uživatele.
- **Kapitola 22 – Komunita Ruby.** Tato kapitola shrnuje důležité webové stránky, e-mailové konference, diskusní skupiny, konference, IRC kanály a další.

V širším smyslu můžeme říci, že každá kapitola v této knize je nová, protože jsem upravil a aktualizoval každou z nich; udělal jsem stovky menších a tucty větších změn. Smazal jsem zastaralé nebo méně důležité věci a změnil materiály tak, aby odpovídaly všem aktuálním změnám v Ruby. Do každé kapitoly jsem samozřejmě přidal nové příklady a komentáře.

Pravděpodobně vás bude zajímat, co všechno bylo přidáno do starých kapitol. Jednou z nejdůležitějších změn je popis enginu Oniguruma, o kterém jsem se již zmínil dříve. Dále musím zmínit popis matematických knihoven a tříd jako `BigDecimal`, `Mathn` a `Matrix`. Nesmím zapomenout ani na nové třídy, jako například `Set` a `DateTime`.

K velkým změnám v obsahu došlo především v následujících kapitolách:

- **Kapitoly 10 – I/O a uložení dat.** Přidány materiály o `readpartial` a neblokujícím I/O. Nechybí materiály o třídě `StringIO`. Také jsem přidal materiály o CSV, YAML a KirbyBase. Do databázové části této kapitoly byly přidány informace o Oracle, SQLite, DBI a diskuse o ORM (Object-Relational Mappers).
- **Kapitola 11 – OOP a dynamické rysy Ruby.** Zahrnuje poslední dodatky k Ruby, například `initialize_copy`, `const_get`, `const_missing` a `define_method`. Do této kapitoly jsem také přidal informace o technice delegování.
- **Kapitola 12 – Grafická prostředí pro Ruby.** Téměř všechno bylo upraveno (zejména sekce o GTK a Fox). Část věnovaná QtRuby je úplně nová.
- **Kapitola 14 – Skriptování a správa systému.** Nově popisuje instalátor Ruby pro Windows a několik podobných balíčků. Došlo také k několika změnám v ukázkových kódech.
- **Kapitola 18 – Síťové programování.** Nově obsahuje část věnovanou e-mailovým přílohám. Interakce se serverem IMAP je rovněž novinkou. Obsahuje popis knihovny `OpenURI`.
- **Kapitola 19 – Ruby a webové aplikace.** Nyní stručně pokrývá Ruby on Rails, Nitro, Wee, IOWA a další webové nástroje. Také pokrývá knihovny `WEBrick` a `Mongrel`.
- **Kapitola 20 – Distribuované Ruby.** Obsahuje nové materiály popisující knihovnu `Rinda`, což je implementace konceptu `tuplespace` v Ruby. Také pokrývá těsně související `Ring`.

Jsou všechny tyto nové informace nezbytné? Ujišťuji vás, že ano.

Kniha *The Ruby Way* byla v pořadí druhou knihu o Ruby, která vyšla v anglickém jazyce. (Jako první vyšla vynikající kniha *Programming Ruby* od Davea Thomase a Andyho Hunta.) Záměrně jsem tuto knihu napsal tak, aby první knihu o Ruby spíše doplňovala, než překrývala. Podle ohlasů to vypadá, že se mi to povedlo.

Když jsem začal psát první vydání knihy o Ruby, neprobíhaly žádné mezinárodní konference o Ruby. Neexistovalo RubyForge, ruby-doc.org nebo rubygarden.org. Stručně řečeno – kromě domovského webu Ruby bylo na internetu velmi málo informací o tomto programovacím jazyku. Archiv aplikací Ruby obsahoval pouze několik stovek položek.

V té době bylo k dispozici pouze několik málo publikací (ať už online nebo offline), které dávaly najevo, že ví o existenci Ruby. Vždy, když vyšel nějaký článek o Ruby, byl to pro nás důvod, abychom si ho všimli a diskutovali o něm na e-mailové konferenci.

Také neexistovaly nástroje a knihovny Ruby, které dnes považujete za zcela běžné. Neexistoval žádný RDoc. Nebyl žádný REXML pro analýzu XML. A matematická knihovna byla – v porovnání se současností – podstatně chudší. Ačkoliv existovala jistá podpora databází, ODBC podporováno nebylo. Tk bylo nejpožívanějším GUI toolkitem a nejběžnější způsob vývoje webových aplikací spočíval na nízkoúrovňové CGI knihovně.

Pro platformu Windows nebyl k dispozici žádný instalátor na jedno kliknutí ("one-click" installer). Uživatelé této platformy obvykle používali Cygwin nebo kompilátor založený na mingw.

Systém RubyGems neexistoval ani v základní formě. Hledání a instalace knihoven a aplikací byl zcela manuální proces, který se neobešel bez zadávání příkazů `tar` a `make`. Nikdo neslyšel o Ruby on Rails, a pokud si dobře pamatuji, nikdo tehdy nepoužíval termín kachní typování (duck typing). Pro Ruby nebyl k dispozici ani YAML, ani Rake.

V té době jsme používali Ruby ve verzi 1.6.4 a mysleli si, jak je úžasná. Ale verzi 1.8.5, kterou používáme dnes, je ještě úžasnější. Ačkoliv došlo k několika změnám v syntaxi, nejedná se o nic důležitého, o čem bychom se měli rozepisovat. Většinou se jedná o okrajové záležitosti, které nyní dávají více smyslu než v minulosti.

Došlo ke změně sémantiky některých základních metod. Opět se většinou jedná o drobné změny. Například `Dir#chdir` dříve nepřijímal blok, v současné době to již umí. Některé základní metody byly zrušeny nebo přejmenovány. Metoda `class` přišla o svůj alias `type`. Metoda `intern` je nyní známa jako metoda `to_sym`. `Array#indices` je nyní `Array#values_at` atd.

Přibýlo také několik nových základních metod, například `Enumerable#inject`, `Enumerable#zip` a `IO#readpartial`. Stará knihovna `futils` se nyní jmenuje `fileutils` a má svůj vlastní modul `FileUtils` se jmenným prostorem namísto přidávání metod do třídy `File`.

Pochopitelně došlo i k jiným změnám, o kterých v této sekci nic nepíši. Nicméně je důležité uvědomit si, že všechny tyto změny byly provedeny s velkou pozorností a opatrností. Ruby je pořád Ruby. Mnoho krásy Ruby je odvozeno z faktu, že všechny změny byly provedeny pozvolna, s rozmyslem a moudrostí Matze a ostatních vývojářů.

A jak to vypadá dnes?

Dnes máme k dispozici více knih o Ruby a více publikovaných článků, než potřebujeme. Web přetéká různými tutoriály, ukázkovými zdrojovými kódy a dokumentacemi. Objevily se nové nástroje a knihovny. Z těchto různých nástrojů se zdají být nejvíce používané webové frameworky, nástroje pro blogování, nástroje pro tvorbu značek a ORM (object-relational mappers). A samozřejmě je

zde k dispozici velké množství dalších – například nástroje a knihovny pro databáze, GUI, náročné výpočty, webové služby, práci s obrázky, správu zdrojů atd.

Podpora jazyka Ruby v editorech je rozšířenější a promyšlenější. IDE jsou velmi užitečné.

Je rovněž nesporné, že komunita Ruby se rozrostla a změnila. Ruby dnes rozhodně není podřadný jazyk – používá ho NASA, NOAA, Motorola a mnoho dalších velkých firem a institucí. Je používán nejenom pro práce s grafikou či databázemi, ale také pro různé výpočty, vývoj webových aplikací a další věci. Stručně řečeno – Ruby jde společně s hlavním proudem.

Aktualizaci této knihy dělám s láskou a věřím, že pro vás bude užitečná.

Jak pracovat s touto knihou

Předpokládám, že Ruby se nebudete učit z této knihy, protože není určena pro úplné začátečníky. Tímto chci říci, že pokud je pro vás Ruby naprostou novinkou, možná bude vhodné začít s nějakou jinou knihou. Nicméně programátoři jsou houževnatá parta, takže pokud máte nějaké zkušenosti s programováním, lze využít tuto knihu pro získání potřebných znalostí o Ruby. V takovém případě vám samozřejmě doporučuji začít kapitolou 1, která obsahuje stručný úvod do Ruby, věci týkající se syntaxe a samozřejmě i několik ukázkových příkladů.

Tato kniha je převážně určena k zodpovězení otázek typu "Jak mohu udělat...?", takže předpokládám, že budete přeskakovat mezi jednotlivými tématy. Ačkoliv bych byl rozhodně poctěn, kdybyste si přečetli každou stránku od začátku až do konce, nevyžaduji to. Spíše očekávám, že budete brouzdat obsahem a hledat techniky, které potřebujete nebo věci, jež jsou pro vás něčím zajímavé.

Od té doby, kdy vyšlo první vydání této knihy, jsem mluvil s mnoha lidmi, a jak se ukázalo, hodně z nich trpělivě četlo tuto knihu stránku po stránce. A co více – několik lidí mi sdělilo, že knihu použili pro učení, takže se ukazuje, že možnosti využití této knihy jsou opravdu velké.

Některé věci v této knize mohou vypadat docela jednoduše a možná se budete sami sebe ptát, proč o nich vlastně píšu. Je to kvůli tomu, že různí lidé mají různé schopnosti a zkušenosti. To, co je samozřejmostí pro jednoho uživatele, nemusí být pochopitelné pro jiného. Tuto knihu lze označit za kompromis, protože ačkoliv mým cílem bylo poskytnout vám komplexní a vyčerpávající informace o Ruby, bylo nezbytné udržet rozsah této knihy na nějaké rozumné úrovni.

Při psaní této knihy jsem předpokládal, že věci, které vás zajímají, budete vyhledávat podle požadované funkcionality nebo vašeho záměru a nikoliv podle názvu metody nebo třídy. Například třída `String` obsahuje několik následujících metod – `capitalize`, `upcase`, `casecmp`, `downcase` a `swapcase`. V nějaké referenční příručce by tyto metody byly zařazeny pod odpovídající písmena abecedy, nicméně v této knize je naleznete pěkně pohromadě.

V honbě za úplností se občas odkazuji na nějaké jiné knihy, kde naleznete další informace. Díky tomu jsem mohl do knihy zařadit více různorodých příkladů a praktických ukázek.

Protože tato kniha je určena programátorům, snažil jsem se do ní dostat co nejvíce okomentovaného kódu. Pokud pominu tento úvod, myslím si, že se mi to podařilo. Ačkoliv jako spisovatel mohu být někdy upovídaný, správný programátor chce vždy vidět kód. (A pokud ne, měl by.)

Některé příklady jsou kompletně vymyšlené, za což se musím omluvit. V některých případech může být problém (nebo zbytečně složitý) ilustrovat požadovanou techniku či princip v kontextu reálného světa. Nicméně – pokud jsem chtěl demonstrovat nějaký komplexnější problém, snažil jsem se vytvořit řešení, které by bylo více založeno na potřebách reálného světa. Takže pokud v knize naleznete téma popisující slučování řetězců, může se vám zdát část kódu, která obsahuje `foo` a `bar` jako nesmyslná. Pokud jsem se ovšem věnoval složitějšímu tématu, jakým je třeba analýza kódu XML, je ukázkový kód mnohem smysluplnější a realističtější.

Tato kniha obsahuje dva nebo tři osobní manýry, ke kterým se předem přiznávám. Jedním z nich je snaha vyhnout se "ošklivým" globálním proměnným ve stylu Perlu, jako například `$_`. Ačkoliv jsou v Ruby přítomny, pracují správně a každodenně jsou používány většinou programátorů v Ruby, téměř vždy se jim můžete vyhnout. A já jsem se rozhodl je vždy vynechat.

Dalším osobním manýrem je to, že se vyhýbám samostatným výrazům, pokud nemají nějaký vedlejší efekt. Ruby je orientováno na výrazy, což je dobrá věc, kterou se v této knize snažím využívat. Ale v ukázkách kódu preferuji nepsat výrazy, které nevrací použitelnou hodnotu. Ačkoliv výraz `"abc" + "def"` může demonstrovat způsob, jakým se slučují řetězce, já místo toho raději napíšu něco jako `str = "abc" + "def"`. Je možné, že toto se vám bude zdát jako zbytečně rozvláčné, nicméně – pokud jste programátorem v jazyce C, který si opravdu všímá toho, zdali jsou funkce platné či nikoliv (nebo pokud jste programátor v jazyce Pascal, jenž přemýšlí v procedurách a funkcích), bude to pro vás mnohem přirozenější.

A poslední věc je ta, že nemám rád znak "mřížka" pro označení instančních metod. Mnoho ze skalních uživatelů Ruby si bude myslet, že jsem ukecaný, když řeknu "metoda instance `crypt` třídy `String`" místo prostého `String#crypt`. Nicméně tohle nikoho nepoplete. (Ve skutečnosti už pomalu začínám přecházet k používání znaku "mřížka", protože je zřejmé, že tato notace nevymizí.)

Vždy, když to bylo možné, snažil jsem se poskytnout odkazy na další zdroje, protože požadavek na rozumný rozsah této knihy mi nedovolil dát do knihy všechno, co jsem chtěl. Každopádně věřím, že vám tyto odkazy pomohou. V knize se například velmi často odkazují na RAA (Ruby Application Archive), což je jeden z nejdůležitějších zdrojů na webu o Ruby, o kterém byste měli vědět.

Úvodní část většiny programátorských knih obvykle obsahuje naprosto zbytečnou ukázkou typografie, které je v knize použita pro výpisy zdrojových kódů (a případně pro další dodatečné informace). V úvodu této knihy nic takového nenaleznete, protože nehodlám urážet vaši inteligenci.

Na závěr chci poukázat na skutečnost, že přibližně 10 procent této knihy bylo napsáno jinými lidmi, čímž nemám na mysli pouze technické úpravy a opravu chyb. Ocením, když si přečtete poděkování v této knize (a samozřejmě i v jakékoli jiné knize). Mnoho čtenářů tuto část knihy bohužel přeskakuje. Běžte si poděkování přečíst teď hned. Prospěje vám to (stejně jako zelenina).

Zdrojové kódy ke stažení

Zdrojové kódy k této knize si můžete stáhnout z adresy zonerpress.cz/download/ruby-kompendium.zip (165 KB). Zdrojový archiv ve formátu `.zip`, obsahuje všechny významné fragmenty kódu. Pro jednotlivé soubory v tomto archivu se používá následující konvence. Výpisy kódu jsou

pojmenovány podle čísel jednotlivých kapitol, například soubor `list11-1.rb` obsahuje kód z výpisu 11.1. Kratší části kódu jsou pak pojmenovány na základě čísla stránky, kde se daný fragment kódu nachází, a nepovinného písmene – například soubory `p240a.rb` a `p240b.rb` se odkazují na dva fragmenty kódu ze strany 240. Části kódu, které jsou příliš krátké, nebo jež nemůžou být spuštěny mimo kontext, se obvykle v archivu neobjevují.

Poznámka redakce k českému vydání

Zdrojové kódy, které jsme pro vás stáhli z adresy www.rubyhacker.com, bohužel nejsou kompletní. Archiv obsahuje výpisy a kratší části kódu pouze do kapitoly 12. Ačkoliv Hal Fulton na své stránce slibuje přidání zdrojových kódů ze zbývajících kapitol do poloviny listopadu 2006, doposud se tak nestalo. Za tuto situaci, kterou není v našich silách ovlivnit, se omlouváme.

Jak jsme napsali již výše, soubory s kratšími částmi kódu jsou v archivu pojmenovány podle čísel stránek, na kterých se nachází. Čísla stránek českého vydání nicméně neodpovídají číslům stránek originálního anglického vydání. Z tohoto důvodu jsme do archivu zahrnuli soubor `ruby-puvodni-obsah.pdf`, který obsahuje obsah z původního anglického vydání, a který můžete použít pro snadnější vyhledání zdrojových souborů s požadovanými fragmenty kódu.

Na závěr tohoto textu chci poděkovat panu Daliborovi Šrámkovi, který odvedl ohromné množství práce při odborných korekturách této knihy. Jeho web naleznete na www.insula.cz/dali/.

Sdělte nám svůj názor

Jako čtenáři této knihy se stáváte těmi nejdůležitějšími kritiky a komentátory. Vážíme si vašeho názoru a chtěli bychom vědět, co děláme správně, co bychom mohli dělat lépe, ve kterých oblastech bychom měli publikovat, a také vaše další podnětné myšlenky, o které jste ochotni se podělit.

Jako odborný redaktor Zoner Press vítám vaše názory. Můžete mi psát – poslat e-mail nebo dopis – a sdělit mi, co se vám v této knize líbilo nebo nelíbilo, stejně tak, co bychom měli udělat, aby naše další knihy byly lepší. Pokud mi napíšete, nezapomeňte, prosím, připojit název knihy, ISBN, jméno autora, vaše jméno, telefon, fax nebo e-mail. Pozorně zhodnotím vaše názory a poskytnu je všem lidem, kteří pracovali na této knize.

Prosím, vězte, že nemohu pomoci s technickými problémy, které se týkají obsahu knihy, a že díky velkému množství e-mailů, jež dostávám, nemohu zaručit odpověď na každou zprávu.

E-mail: miroslav.kucera@zoner.cz nebo knihy@zoner.cz.

Adresa: ZonerPress,

ZONER software, s.r.o.,

Miroslav Kučera,

Nové sady 18,

602 00 Brno.

Co je "cesta Ruby"?

Nechte nás připravit se na zápas s nepopsatelným, a uvidíte, možná se na to nakonec nevykašleme.

– Douglas Adams, Holistická detektivní kancelář Dirka Gentlyho

Jak si představit cestu Ruby (Ruby Way)? Věřím, že má dva aspekty. První je filozofie návrhu Ruby; druhý je filozofie jeho použití. Je přirozené, že návrh a použití spolu obvykle souvisí, ať už se jedná o software nebo hardware. Jestliže postavím nějaké zařízení a nasadím na něj kliku, je to proto, že očekávám, že ji někdo uchopí.

Ruby má jakousi nepojmenovanou kvalitu, která ho dělá tím, čím je. Vidíme tuto kvalitu v návrhu syntaxe a sémantiky jazyka, ale i v programech napsaných pro interpret jazyka. Jenže jakmile nakreslíme tuto dělicí čáru, hned se nám rozmaže.

Ruby zjevně není pouze nástroj pro tvorbu softwaru, ale sám o sobě je to také kus softwaru. Proč by programy v Ruby měly pracovat podle odlišných pravidel, než pracuje interpret? Koneckonců, Ruby je vysoce dynamický a rozšiřitelný jazyk. Mohou existovat důvody, proč by se tyto dvě úrovně měly tu a tam odlišovat; je to výhodné pro přizpůsobení se problémům skutečného světa. Ale z obecného pohledu mohou být (a měly by být) myšlenkové pochody stejné. Ruby může být implementováno v Ruby, ve vskutku Hofstadterově stylu, i když v době psaní knihy tomu tak není.

Ačkoliv často nepřemýšlíme nad původem slova "cesta" (way), toto slovo má dva různé významy, ve kterých se používá. Na jedné straně to znamená metodu, způsob nebo techniku, na straně druhé pak může znamenat cestu nebo stezku. Je jasné, že oba tyto významy spolu vzájemně souvisejí a když říkám "cesta Ruby" (the Ruby Way), míním tím oba významy.

To, o čem tady mluvíme, je nejenom proces myšlení, ale také cesta, kterou následujete. Ani největší softwarový guru nemůže tvrdit, že dosáhl dokonalosti. Může pouze tvrdit, že následoval cestu. A ačkoliv zde může být více než jedna cesta, já zde mohu mluvit pouze o jedné.

Tradiční moudrost říká, že forma následuje funkci. A tradiční moudrost má obvykle pravdu. Ale Frank Lloyd Wright jednou řekl: "Forma následuje funkci – to je nepochopeno. Forma a funkce by měly být ve shodě, duchovně spojeny." Co tím Wright myslel? Myslím, že tuto pravdu se nelze naučit z knihy, ale pouze ze zkušenosti.

Nicméně, chtěl bych poukázat na to, jakým způsobem Wright vyjádřil tuto pravdu v trochu stravitelnější podobě. Byl velký zastánce jednoduchosti, jednou dokonce řekl, že "nejužitečnějšími nástroji architekta jsou guma na rýsovacím prkně a páčidlo na staveništi".

Jednou ze ctností Ruby je jednoduchost. Mám k tomuto tématu citovat další myslitele? Podle Antoine de St. Exupéryho: "Dokonalosti je dosaženo nikoliv tehdy, když není co přidat, ale když není co odebrat".

Ruby je ovšem složitý jazyk. Jak tedy mohu tvrdit, že je jednoduchý? Pokud lépe porozumíme vesmíru, možná nalezneme "zákon zachování složitosti" – fakt reality, který narušuje naše životy jako entropie, jíž se nemůžeme vyhnout, ale pouze ji přerozdělit. A tohle je klíč. Složitosti se nemůžeme

vyhnout, nicméně ji můžeme nechat okolo. Můžeme ji pohřbit mimo dohled. Tohle je starý princip "černé skříňky", která vykonává složité úkoly, nicméně zvenčí je ovládána jednoduchými příkazy.

Pokud jste ještě neztratili trpělivost s mými citacemi, pak je vhodné uvést citát Alberta Einsteina "Všechno by mělo být tak jednoduché, jak to jenom jde, ale ne jednodušší".

V Ruby z pohledu programátora vidíme jednoduchost (když ne z pohledu těch, kdo spravují samotný interpret). Vidíme také potenciál pro kompromis. Ve skutečném světě se musíte trochu ohýbat. Například každá entita v Ruby by měla být opravdovým objektem, nicméně určité hodnoty (jako například celá čísla) jsou uloženy jako přímé hodnoty. V obchodu, který je studentům počítačové vědy důvěrně znám po desetiletí, jsme vyměnili praktičnost implementace za eleganci návrhu. V důsledku jsme vyměnili jeden druh jednoduchosti za jiný.

To, co Larry Wall řekl o Perlu, je pravda: "Když něco řeknete v malém jazyce, zdá se to velké. Když něco řeknete ve velkém jazyce, zdá se to malé". Totéž platí pro angličtinu. Hlavní důvodem, proč biolog Ernst Haeckel mohl pouze třemi slovy sdělit, že "ontogeneze rekapituluje fylogenezi", bylo to, že měl k dispozici tato silná slova se specifickým významem. Povolujeme vnitřní složitost jazyka, protože nám umožňuje odsunout tuto složitost do pozadí v jednotlivých výrocích.

Řečeno jinými slovy – nepište 200 řádků kódu, když 10 řádků udělá stejnou práci. Stručnost je dobrá věc. Krátká část programu zabere v mozku programátora méně místa a lze ji snadněji uchopit jako celek. A jako pozitivní vedlejší efekt se do kódu při psaní zanesou méně chyb.

Samozřejmě je stále potřeba myslet na Einsteinovo varování o jednoduchosti. Pokud máte stručnost ve svém žebříčku priorit hodně vysoko, skončíte s kódem, který bude beznadějně nečitelný. Informační teorie říká, že komprimovaná data mají podobné statistické vlastnosti jako náhodný šum. Pokud se díváte na C, APL nebo na notaci (špatně napsaného) regulárního výrazu – máte právě tento dojem. "Ano, jednoduše, ale ne příliš jednoduše." Tohle je klíč. Stručně, ale nikoliv na úkor čitelnosti.

Stručnost a čitelnost jsou dobré. Ale to má společnou příčinu, na kterou často zapomínáme. A sice, že počítače existují pro lidi, nikoliv lidi pro počítače. Dříve to bylo naopak. Počítače stály miliony dolarů a spotřebovaly velké množství energie. Lidé se chovali, jako by počítač byl ztělesněním boha; programátoři se považovali za prosebníky. Čas počítače byl cennější než čas člověka. Když se počítače staly menšími a levnějšími, staly se velmi populárními vysokoúrovňové jazyky. Ačkoliv něco takového bylo zcela zbytečné z hlediska počítačů, bylo to velmi efektivní z pohledu lidského. Ruby je pouze dalším rozvojem těchto myšlenek. Někdo občas použije zkratku VHLL (Very High-Level Language). Ačkoliv tento termín není přesně definován, myslím si, že jeho použití zde je opodstatněné.

Počítač je předurčen k tomu, aby byl náš sluha, nikoliv pán. A jak Matz s oblibou často říká: chytrý sluha by měl vykonat složité úkoly prostřednictvím několika krátkých příkazů. To je správná myšlenka pro celou historii počítačové vědy. Na začátku byl strojový kód, přičemž postupně jsme přecházeli k nízkoúrovňovým a poté i k vysokoúrovňovým jazykům. Mluvím zde o posunu od formy zaměřené na stroj k formě zaměřené na člověka. Podle mého názoru je Ruby výborným příkladem programování orientovaného na člověka.

Přesuňme se do jiné doby. V roce 1980 vyšla nádherná malá kniha s názvem *The Tao of Programming* (napsal ji Geoffrey James). Téměř každý řádek této knihy stojí za zmínku, ale já zopakuji pouze jednu věc. Program by měl dodržovat "pravidlo nejmenšího údivu". Co to znamená? Nic víc než to, že program by měl vždy reagovat na uživatele takový způsobem, aby ho co nejméně překvapil. (V případě interpretu jazyka je uživatelem samozřejmě programátor.) Ačkoliv si nejsem zcela jist, zdali lze Geoffreymu Jamesovi připsat autorství tohoto pravidla, v jeho knize jsem se s ním setkal poprvé. V komunitě Ruby se jedná o velmi známé a velmi často citované pravidlo. Obvykle se ovšem nazývá jako princip nejmenšího překvapení (principle of least surprise, POLS).

Každopádně, ať už toto pravidlo nazýváte jakkoliv, rozhodně platí a bylo dokonce základní direktivou v průběhu vývoje jazyka Ruby. Je také užitečné pro ty, co vyvíjejí různé knihovny nebo uživatelská rozhraní. Jediným problémem je samozřejmě to, že různí lidé mohou být překvapeni různými věcmi; neexistuje žádná univerzální dohoda, jak by se měl nějaký objekt nebo metoda chovat. Matz říká, že "nejmenší překvapení" by se především mělo vztahovat na něj, protože on je návrhář jazyka. Čím více budete myslet jako on, tím méně vás Ruby překvapí. A ujišťuji vás, že napodobování Matze není špatný nápad pro spoustu z nás.

Nezávisle na tom, jak logicky je systém vybudován, vaše intuice potřebuje trénink. Každý programovací jazyk je svět sám pro sebe, se svými vlastními předpoklady. Totéž platí pro lidské jazyky. Když jsem se učil němčinu, dozvěděl jsem se, že všechna podstatná jména se píší s prvním písmenem velkým kromě slova *deutsch*. Postěžoval jsem si na to svému profesorovi, konec konců, vždyť je to *název* jazyka, nebo ne? Usmál se a řekl: "Nebojuj s tím."

To, co mě naučil, bylo, abych nechal němčinu němčinou. To je dobrá rada pro každého, kdo k Ruby přichází od nějakého jiného jazyka. Nechte Ruby, aby mohlo být Ruby. Nečekejte, že je to Perl, protože není. Nečekejte, že je to LISP nebo Smalltalk, také není. Na druhé straně Ruby obsahuje společné prvky se všemi třemi zmíněnými. Postupujte tedy podle svých očekávání, ale nebojujte s tím, že někdy nebudou splněna. (Pokud Matz neodsouhlasí, že se jedná o potřebnou změnu.)

Každý programátor dnes zná princip ortogonalit (orthogonality). Předpokládejme, že máme fiktivní dvojici os se sadou srovnatelných jazykových entit na jedné a s vlastnostmi nebo schopnostmi na druhé. Když řekneme slovo ortogonalita, obvykle tím myslíme, že prostor definovaný těmito osami, je tak "plný", jak jen to lze logicky udělat.

Cesta Ruby (Ruby way) se částečně snaží o ortogonalitu. Pole je v některých věcech podobné jako haš, protože operace na každém z nich mohou být podobné. Limitu je ovšem dosaženo, jakmile vstoupíte do oblasti, kde se tyto dvě věci odlišují. Matz říká, že "přirozenost" je hodnocena více než ortogonalita. Ale pro porozumění toho, co je přirozené a co nikoliv, musíme přemýšlet a psát kód.

Ruby se snaží být k programátorovi maximálně přátelský. Například jsou k dispozici aliasy pro velké množství metod – jak `size`, tak `length` vrátí počet jednotek pole. Pravopisně odlišné `indexes` a `indices` se obě odkazují na stejnou metodu. Ačkoliv někdo tohle považuje za komplikaci nebo rovnou za špatnou vlastnost, mně se to docela líbí.

Ruby dále usiluje o konzistenci a pravidelnost. Není na tom nic tajemného. V každém aspektu života toužíme po tom, aby věci byly pravidelné a vzájemně podobné. Obtížnější na tom je nau-

čit se, kdy tyto principy porušit. Například Ruby má ve zvyku ke jménům predikátových metod připojovat otazník (?). To je správné a dobré, protože to dělá kód přehlednějším a jmenný prostor snadněji ovladatelným. Ale podstatně diskutabilnější se může zdát obdobné použití vykřičníku (!) pro označení metod, které jsou "destruktivní" nebo "nebezpečné" v tom smyslu, že modifikují příjemce. Je to diskutabilní z toho důvodu, že tímto způsobem nejsou označeny úplně všechny destruktivní metody. Neměli bychom být konzistentní?

Ne, skutečně bychom neměli. Některé z metod svého příjemce mění přirozeně (jako například `Array` metody `replace` a `concat`). Některé metody umožňují přiřazení do vlastnosti třídy, takže bychom neměli přidávat vykřičník k názvům vlastností nebo ke znaménku rovnosti. Některé metody pravděpodobně mění příjemce, jako třeba metoda `read` – to by zase znamenalo příliš časté použití tohoto značení. Kdyby název každé destruktivní metody končil vykřičníkem, naše programy by brzy vypadaly jako reklamní brožury pro multi-level marketing.

Povšimli jste si jistého napětí mezi protichůdnými silami? Tendence porušit někdy každé pravidlo? Dovolte mi tohle zformulovat jako druhý Fultonův zákon: "Každé pravidlo má výjimku, kromě druhého Fultonova zákona". (Ano, jedná se o žertík, ale jen malý.)

To, co vidíme v Ruby, není hloupé uplatňování konzistence. Není to ani přísné lpění na jednoduchých pravidlech. A vskutku – částí cesty Ruby je to, že se nejedná o přísný a nepoddajný přístup. V návrhu jazyka, jak jednou řekl Matz, byste měli "následovat své srdce".

Dalším aspektem filozofie Ruby je toto – neobávejte se změn za běhu programu a nemějte strach z dynamických věcí. Celý svět kolem vás je dynamický; proč byste měli programovat staticky? Ruby lze označit za jeden z nejvíce dynamických jazyků v historii.

Chtěl bych také zmínit následující aspekt – nebuďte otrokem výkonu. Pokud je výkon aplikace nepřijatelný, problém musí být pochopitelně vyřešen, ale za normálních okolností by to neměla být první věc, nad kterou budete přemýšlet. Pokud výkonnost není kritická, preferujte eleganci nad výkonností. Avšak pokud píšete nějakou knihovnu, která může být používána nepředvídatelnými způsoby, výkon může být rozhodující od počátku.

Když se podívám na Ruby, všímám si rovnováhy mezi odlišnými cíli návrhu, složité interakce připomínají problém mnoha těles ve fyzice. Umím si docela dobře představit, že to může být formováno jako díla Alexandra Caldera. Je to pravděpodobně tato interakce sama o sobě, harmonie, kterou ztělesňuje filozofie Ruby, spíše než její individuální části. Programátoři vědí, že jejich řemeslo není jen věda a technologie, ale i umění. Váhám, zdali říci, že v informatice existuje nějaký spirituální aspekt, ale jen tak mezi námi, pravděpodobně ano. (Pokud jste nečetli knihu *Zen and the Art of Motorcycle Maintenance*, kterou napsal Robert Pirsig, doporučuji vám to udělat.)

Programovací jazyk Ruby vyvstal z lidského nutkání vytvořit věc, která bude užitečná a krásná. Program napsaný v Ruby by měl pramenit z toho samého Bohem daného zdroje. To je pro mě podstata cesty Ruby.

