„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

kompendium znalostí pro začátečníky i profesionály

Hal Fulton

Addison-Wesley

Zoner PRESS

THE RUBY WAY, SECOND EDITION
Hal Fulton

Ruby – kompendium znalostí pro začátečníky i profesionály

Autor: Hal Fulton

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`

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

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 nejjednoduššího 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 programátoru 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 filozofií 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 regulárních 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 prvé 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 potíž 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.

Zbývající nové kapitoly 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 kapitolach:

- **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 nejpoužívanějším 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 současně 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.

Přibylo také několik nových základní 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 složité) 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 nedovolit dát do knihy všechno, co jsem chtěl. Každopádně věřím, že vám tyto odkazy pomohou. V knize se například velmi často odkazuji 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 ukázku 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 jakékoliv 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 stáhnuli 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 dělící 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. Exuperyho: "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 pohřbít 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 zanese 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 ortogonality (orthogonality). Předpokládejme, že máme fiktivní dvojici os se 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 i `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 modifikuji 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.