Ruby

Jarne Peters

February 3, 2024

Ruby

- 1995 von Yukihiro Matsumoto entwickelt.
- Motivation: Kombination der besten Features aus Perl, Smalltalk, Lisp und Python.
- Einzigartig durch Dynamik und Interpretierbarkeit.
- Elegante Organisation von Daten und Funktionen in Form von Objekten.
- Ruby ist vor allem wegen seiner einfachen Nutzung beliebt.
- Yukihiro Matsumoto legte besonders großen Wert darauf, dass Programmieren mit Ruby Spaß macht.

Ruby Syntax

- Die Syntax von Ruby ähnelt der von Perl oder Python.
- Klassen und Methoden werden durch Schlüsselwörter gekennzeichnet.
- Variablen in Ruby benötigen keinen Sigil als Präfix (im Gegensatz zu Perl).
- Ruby zeichnet sich durch Einfachheit und Ausdrucksstärke aus.
- Typen werden dynamisch interpretiert.

Blöcke in Ruby

- Blöcke in Ruby ermöglichen die Erstellung anonymer Funktionen und deren Weitergabe an Methoden.
- Sie können durch das ‘do..end‘-Statement oder geschweifte Klammern ‘{}‘ definiert werden.
- Die Verwendung von Blöcken führt zu kompaktem und wiederverwendbarem Code.
- Das Schlüsselwort ‘yield‘ ruft den Code innerhalb des Blocks auf

```ruby
def print_hello
  yield
  end

  print_hello { puts "Hello" }
```

Ruby Mixins

- In Ruby keine Mehrfachvererbung, stattdessen Module als Mixins.
- Mixins werden in Klassen eingefügt, um Funktionalitäten zu teilen, ohne direkte Vererbung.
- Im Gegensatz zur klassischen Vererbung erlaubt Ruby, mehrere Module in eine Klasse einzufügen.
- Reduzieren Redundanz, ermöglichen flexible Hinzufügung von Funktionen ohne komplexe Vererbungshierarchien.
- Leistungsstarkes Feature für Wiederverwendbarkeit und Flexibilität ohne Beeinträchtigung der klaren Struktur.

Ruby Mixins

```ruby
module Debug
  def whoAmI?
    "#{self.type.name} (\##{self.id}): #{self.to_s}"
  end
end
class Phonograph
  include Debug
  # ...
end
class EightTrack
  include Debug
  # ...
end
ph = Phonograph.new("West End Blues")
et = EightTrack.new("Surrealistic Pillow")
ph.whoAmI?  »   "Phonograph (#537766170): West End Blues"
et.whoAmI?  »   "EightTrack (#537765860): Surrealistic Pillow"
```

Entwicklung von Ruby

- Als Open-Source-Programmiersprache ermöglicht Ruby Entwicklern weltweit, am Code mitzuwirken und Verbesserungen vorzuschlagen.
- Der Quellcode von Ruby wird auf GitHub gehostet.
- Interessierte können sich der Mailingliste von Ruby anschließen, um aktiv an der Entwicklung teilzunehmen.
- Die Mailingliste teilt wichtige Informationen, Diskussionen und Entscheidungen zur Weiterentwicklung der Sprache.
- Die Veröffentlichung von Ruby-Updates erfolgt in der Regel jährlich um die Weihnachtszeit.

Dokumentation von Ruby

- Die Ruby-Dokumentation ist die beste Ressource für Entwickler, um die Sprache effektiv zu lernen, zu verstehen und zu nutzen.
- Die offizielle Website von Ruby bietet eine umfassende Dokumentation sowohl für Anfänger als auch für erfahrene Entwickler.
- Neben grundlegenden Informationen bietet die Website detaillierte Anleitungen, Tutorials und Referenzmaterialien.

RubyGems: Package Management in Ruby

- RubyGems stellt ein standardisiertes Format für die Bereitstellung von Bibliotheken und Programmen namens Gems bereit.
- Dies stellt sicher, dass Gems konsistent formatiert sind und ermöglicht einen reibungslosen Austausch zwischen verschiedenen Projekten und Entwicklern.
- Gems werden mit dem Gem-Befehlszeilentool installiert
- Entwickler können mit diesen erforderliche Bibliotheken herunterladen, installieren und in ihre Projekte integrieren
- RubyGems ist seit Ruby Version 1.9 in der Standard-Ruby-Distribution enthalten.

Andere Ruby-Implementierungen

- Ruby hat nicht nur eine Implementierung, sondern viele.
- Die Standardimplementierung wird als MRI (Matz's Ruby Interpreter) oder Cruby bezeichnet
- Es gibt alternative Implementierungen mit verschiedenen Features und Funktionen.
- JRuby basiert auf der Java Virtual Machine (JVM)
- Rubinius ist in Ruby selbst geschrieben und baut auf LLVM auf
- Truffle Ruby ist eine leistungsstarke Implementierung
- Jede Implementierung bietet einzigartige Vorteile und Funktionen, die die Vielfalt der Ruby-Entwicklung bereichern.

Ruby Interpreter

- Ruby ist eine interpretierte Programmiersprache.
- Erste Generation von Ruby-Interpretern war MRI (Matz's Ruby Interpreter) bis Version 1.9.
- MRI wurde von Yukihiro Matsumoto in C geschrieben, war aber aufgrund des Global Interpreter Lock (GIL) nur begrenzt für parallele Programmierung geeignet.
- In Version 1.9 wurde MRI durch YARV (Yet Another Ruby VM) ersetzt, der einen JIT-Compiler einführte und die Codeausführung beschleunigte.
- GIL bleibt erhalten, wodurch echte parallele Ausführung von Ruby-Threads nicht möglich ist.

Ruby on Rails

Entwicklung und Einführung:

- Ruby on Rails (Rails) ist ein Web Application Framework, entwickelt in der Programmiersprache Ruby.
- Veröffentlicht 2004, schnell an Beliebtheit gewonnen durch innovative Features.

Model-View-Controller (MVC)-Muster:

- Grundlegendes Konzept, strukturiert Anwendungen in Modell, Controller und Ansicht.
- Modell repräsentiert Daten und Beziehungen, Controller handhabt Anfragen, Ansicht visualisiert Daten.
- "Don't Repeat Yourself" (DRY): Informationen nur einmal in der Datenbank, fördert Code-Wiederverwendbarkeit.
- "Convention over Configuration": Sinnvolle Standardwerte, minimiert Konfiguration, spart Entwicklungszeit.

Ruby on Rails

- Fokus auf der Freude am Programmieren, Spaß bei der Entwicklung von Anwendungen.
- Nach dem "Bigger Smile" Prinzip gestaltet, um Entwicklern ein positives Erlebnis zu bieten.
- Plattformen wie GitHub, Shopify und Twitch nutzen Ruby on Rails.

Grep Clone Implementation

- Umsetzung eines Grep-Klons in drei Hauptabschnitten.
- Optionen, Matching und parallele Ausführung.
- Verwendung der OptionParser-Bibliothek für die Definition von Befehlszeilenoptionen.
- Farbliche Hervorhebung von Treffern durch Integration der Colorize-Bibliothek.
- Hauptfunktion {search_file} verarbeitet Dateien in Chunks, minimiert Speicherverbrauch.
- Parallelisierung mit der Parallel-Bibliothek für effiziente Ausführung.

Grep Clone Implementation

- Suche nach regulären Ausdrücken mit der {match}-Methode
- Leistungsanalyse mit dem Benchmark-Gem und Identifizierung von Verbesserungsmöglichkeiten.
- Herausforderungen waren Leistungsproblemen und Einschränkungen des MRI-Interpreters.
- Klar strukturierte Syntax und hohe Lesbarkeit erleichtern den Entwicklungsprozess.

Grep Clone Implementation

```ruby
options = { after_context: 0, before_context: 0 }
OptionParser.new do |opts|
    opts.banner = "Usage: searcher [OPTIONS] PATTERN [PATH ...]"
    opts.on("-A", "--after-context LINES", Integer, "Prints the given number of following lines for each match") do |a|
        options[:after_context] = a
    end
    opts.on("-B", "--before-context LINES", Integer, "Prints the given number of preceding lines for each match") do |b|
        options[:before_context] = b
    end
    opts.on("-c", "--color", "Print with colors, highlighting the matched phrase in the output") do |c|
        options[:color] = true
    end
    opts.on("-C", "--context LINES", Integer, "Prints the number of preceding and following lines for each match. This is equivalent to setting --before-context and --after-context") do |c|
        options[:before_context] = c
        options[:after_context] = c
    end
    opts.on("-h", "--hidden", "Search hidden files and folders") do |h|
        options[:hidden] = h
    end
    opts.on("-i", "--ignore-case", "Search case insensitive") do |i|
        options[:ignore_case] = true
    end
    opts.on("--no-heading", "Prints a single line including the filename for each match, instead of grouping matches by file") do |nh|
        options[:no_heading] = true
    end
    opts.on("--help", "Print this message") do
        puts opts
        exit
    end
end.parse!(into: options)
```

Grep Clone Implementation

```ruby
def search_file(file, pattern, options)
    chunk_size = 400000
    line_number = 0
    line_buffer = []
    printed_lines = {}
    context_before = []
    context_after = []
    filename = file




    if options[:ignore_case] && pattern.include?('\w') && pattern.include?('\s')
        pattern = pattern.encode('UTF-8').downcase
        pattern_without_w = pattern.gsub('\w', '[\p{L}\p{N}_à]')

        pattern_regex = Regexp.new(pattern_without_w)

    else
        pattern = pattern.encode('UTF-8')
        pattern_without_w = pattern.gsub('\w', '[\p{L}\p{N}_à]')
        pattern_regex = options[:ignore_case] ? Regexp.new(pattern_without_w.to_s, Regexp::IGNORECASE) : Regexp.new(pattern_without_w)


    end
```

Grep Clone Implementation

```ruby
if options[:before_context]
if line_number - options[:before_context]-1 > last_match_line.to_i
    puts "--"
end
context_before.each_with_index do |context_line, i|
    linen = line_number - context_before.size + i
    output_line = "#{file}-#{line_number - context_before.size + i}-#{context_line}"
    unless printed_lines[output_line]||context_line.match(pattern_regex)||printed_lines[linen]
        puts output_line
        printed_lines[output_line] = true
    end
end
end
```

Grep Clone Implementation

```ruby
if File.directory?(path)
    files = Dir.glob("#{path}/**/*", File::FNM_DOTMATCH).reject { |file| File.directory?(file) || (!options[:hidden] && File.basename(file).start_with?('.')) }
else
    files = [path]
end

num_processes = Etc.nprocessors
Parallel.each(files, in_processes: num_processes) do |file|
    search_file(file, pattern, options)
end
```

Fazit: Die Vielseitigkeit und Zukunft von Ruby

- Ruby hat sich als vielseitig und einflussreich in verschiedenen Anwendungsbereichen bewiesen.
- Besonderer Fokus auf Webentwicklung durch das beliebte Ruby on Rails-Framework.
- Einfachheit und Lesbarkeit von Ruby-Code erleichtern den Einstieg und effizientes Programmieren.
- Ruby wird voraussichtlich eine bedeutende Rolle in der Softwareentwicklung beibehalten.
- Herausforderungen durch zunehmende Verwendung von Ruby on Rails im Vergleich zu anderen Frameworks.

Quellen

- https://www.ruby-lang.org/
- https://rubyonrails.org/
- https://www.jetbrains.com/de-de/lp/devecosystem-2021/ruby/
- https://ruby-doc.com/docs/ProgrammingRuby/html/tut_modules.html
- https://ruby-doc.com/docs/ProgrammingRuby/html/tut_modules.html
- https://rubygems.org