Ruby (Programmiersprache)

Wechseln zu:Navigation, Suche

Inhaltsverzeichnis

1 Definition

Ruby ist eine objektorientierte Skriptsprache, die Syntaxelemente von Perl mit Features von Smalltalk kombiniert.

2 Geschichte

Ruby wurde in Japan von Yukihiro "Matz" Matsumoto entwickelt und 1995 in der ersten Version veröffentlicht.

3 Interpreter

Abgesehen vom Original-Interpreter MRI (Matz-Ruby-Interpreter) gibt es inzwischen Portierungen für Java (JRuby) oder .NET (IronRuby). Damit lassen sich mit Ruby alle Klassen und Bibliotheken der zugrunde liegenden Plattform verwenden.

4 Integration

Die Integration von JRuby geht sogar so weit, dass sich mit JRuby on Rails JEE-konforme Webanwendungen entwickeln lassen, die man in einem JEE-Container deployen kann. Noch bessere Bytecode-Kompatibilität für Java bietet Groovy.

5 Features

plattformübergreifend verwendbar
strenges objektorientiertes Paradigma: Alle Datentypen sind Objekte, auch primitive Datentypen wie

Integer. Mehr noch: Sogar Klassen sind Klassenobjekte.
Unterstüzung von regulären Ausdrücken auf Sprachebene
Automatisches Garbage Collection
Metaprogrammierung: Klassen und Objekte können zur Laufzeit "geöffnet" und Methoden hinzugefügt oder verändert werden.
Mixins: Module können in Klassen "eingemischt" werden.
Packetmanager RubyGems

6 Ruby Shell

Bei jeder Ruby-Installation ist eine Shell enthalten. Die „irb“-Shell („Interactive Ruby“) kann verwendet werden, um schnell und einfach Ausdrücke zu testen.

7 Beispiele

Integer:

```
irb(main):001:0> n = 2 * 3 + 2
=> 8
irb(main):002:0> n.class
=> Fixnum
```

String:

```
irb(main):001:0>n = "Hallo " * 3
=> "Hallo Hallo Hallo"
irb(main):002:0> n.class
=> String
irb(main):003:0> n[0,5]
=> "Hallo
```

Schleifen:

```
irb(main):001:0> a = 0
=> 0
irb(main):002:0> 1.upto(5) do |i| a = a + i end
=> 1
irb(main):003:0> a
=> 15
```

Listen und Iterator:

```
irb(main):001:0> a = [ "a", "b", 2, "z", "21" ]
=> ["a", "b", 2, "z", "21"]
irb(main):002:0> a.pop
=> "21"
irb(main):003:0> a
=> ["a", "b", 2, "z"]
irb(main):004:0> a.each do |c| puts c.class end
String
String
Fixnum
String
=> ["a", "b", 2, "z"]
```

8 Metaprogrammierung, Codebeispiele

Metaprogrammierung, Klassenerweiterung:

```ruby
class Auto
  def gas_geben
    "gas geben"
  end
end

# Die Klasse Auto wird geöffnet und die Methode gas_geben neu definiert
class Auto
  def gas_geben
    "noch schneller gas geben"
  end
end

a = Auto.new
puts a.gas_geben # => "noch schneller gas geben"
```

Metaprogrammierung, Objekterweiterung:

```ruby
class Auto
  def gas_geben
    "gas geben"
  end
end

# Die Klasse Auto wird geöffnet und die Methode gas_geben neu definiert
class Auto
  def gas_geben
    "noch schneller gas geben"
  end
end


a = Auto.new

# Hier wird nur dem Objekt a eine Methode hinzugefügt
class << a
  def bremsen
    "bremsen..."
  end
end

puts a.bremsen # => "bremsen..."
puts Auto.new.bremsen # => noMethodError
```

Metaprogrammierung mit class_eval

```ruby
class Auto
  def gas_geben
    "gas geben"
  end
end


class Auto
  def gas_geben
    "noch schneller gas geben"
  end
end


a = Auto.new

class << a
  def bremsen
    "bremsen..."
  end
end

# Ab sofort gibt es für Auto-Klassen auch eine bremsen-Methode
Auto.class_eval do
  def bremsen
    "quieeeetscch......"
  end
end


puts a.bremsen # => "bremsen..."
puts Auto.new.bremsen # => "quieeeetscch......"
```

class_eval evaluiert einen übergebenen Block oder String im Klassenkontext.

```ruby
method_name = "bremsen"
Auto.class_eval <<-END
  def #{method_name}; "quieetsch....."; end;
END
```

class_eval mit übergebenem String

Meta-Programmierung: Die Hookmethode method_missing:

```
class Bike
  def method_missing(name, *args)
    puts "die methode #{name} mit den parametern #{args.join(", ")}
existiert nicht."
  end
end

b = Bike.new
puts b.rahmentypen "carbon", "edelstahl"
```

method_missing wird aufgerufen wenn eine Methode nicht existiert.

JRuby: Verwendung von Swing mit JRuby

```
require "java"
frame = javax.swing.JFrame.new()
frame.setSize 200, 200
frame.add javax.swing.JLabel.new("hallo welt")
frame.setVisible true
```

9 Quellen

Ruby-Website
Armin Roehrl und Stefan Schmiedl (2004): Big in Japan. Linux-Magazin Sonderheft Scripting Edition, Linux New Media AG
JRuby
Groovy
[1] Pickaxe-Buch auf deutsch

10 Siehe auch

http://www.rubyenterpriseedition.com/
http://tryruby.hobix.com/ #Try Ruby in your browser

Dieser Artikel ist GlossarWiki-konform.

In diesem Artikel sollten die Quellenangaben überarbeitet werden.
Bitte die Regeln der GlossarWiki-Quellenformatierung beachten.

Kategorie:

Ruby