Ruby
Version 2006-07-29

Dirk Meyer
dinoex@dinoex.sub.org.org

Schon wieder einen neue Programmiersprache? Haben wir den immer noch nicht genug davon?
Nun Ruby ist anders, und es lohnt sich mehr als nur einen Blick darauf zu werfen.

1. Was ist Ruby?

Ruby ist eine Skript-Sprache, d.h. Sie wird zur Laufzeit interpretiert. Daher ist kein eigener Compiler erforderlich. Ruby ist vollständig o bjektorientiert, und unterstützt den Programmierer in vielerlei. Hinsicht. Viele Sprachen nennen sich zwar "Objektorientiert", aber nur wenig haben mehr als nur die primitivsten Elemente realisiert.

2. Ist Ruby neu?

Nein, der Erfinden "Yukihiro Matsumoto" hat Ruby im Jahre 1993 veröffentlicht und seitdem wird Ruby weiterentwickelt.
Der Namens-Pate war der Edelstein "Rubin"

3. Wofür ist Ruby?

Ruby ist sehr dazu geeignet schnell und einfach funktionsfähige Programme zu erstellen, von der einfachen Verarbeitung von Text bis zu systemnahen Aufgaben. Die Sprache ist leicht erweiterbar und zudem auf fast allen Betriebssystemen zu Hause:
Linux, UNIX, DOS, MS-Windows, MacOS, BeOS, OS/2, etc.

4. Ist Ruby offen?

Ruby ist kostenlos, und frei, jeder darf Ruby kopieren, ändern und sogar weitergeben.

5. Was kann Ruby?

5.1. Vollständig Objektorientiert

Ruby ist als reine objektorientierte Sprache entworfen, Alle Daten in Ruby sind Objekte, es gib keine Ausnahme. Das ist konsequent genau so wie in der Sprache "Smalltalk". Er wurde sehr darauf geachtet das alles rund zueinander passt, aber das System trotzdem erweiterbar ist. In einem Ruby Programm ist daher zur Laufzeit alles änderbar. Objekte, Methoden und Klassen können je nach Problem leicht angepasst werden. Zudem kennt Ruby "Module" als Gruppe von Methoden und Eigenschaften, so kann man seine eigenen Daten schnell mit leistungsfähigen Klassen kombinieren. Die Feature-Liste von Ruby ist lang.

Ein Beispiel in der Praxis zeigt dies besonders. Die Aufgabe ist extrem einfach, wir addieren Zahlen, so benötigt ein DTAUS Datensatz die Summer der Kontonummern als Schutz gegen Lese-

Fehler. Wenn das Ergebnis nicht mehr in ein Wort passt, trennt sich die Spreu vom Weizen. Die meisten Sprachen generieren einen Fehler, Perl und AWK nehmen halt eine Gleitkommazahl. Dumm, denn das ist keine richtige Lösung, und wir müssen viel Aufwand in die Bearbeitung dieser Sonderfälle stecken. Ruby ist da schlauer! Bei einer solchen Operation ändert sich der Typ des Objektes von einem einfachen "Fixnum" in ein "Bignum" und das Programm kann seine Aufgabe ohne einen einzige Änderung vollständig durchführen.
In den Falle wo das Ergebnis kleiner wird, erzeugt Ruby natürlich auch wieder den kleineren Objekt-Typ. Das Programm bleibt trotz unterschiedlichen Daten fehlerfrei.

5.2. Eine einfache Syntax

Wenige Elemente mache die Programme überschaubarer. Operatoren sind nur Zugabe, man kann sie leicht neu definieren. Klammen sind oft nicht erforderlich. Der Name einer variable entscheidet über Ihren Gültigkeitsbereich, z.B. alle Globalen Variablen beginnen mit dem Zeichen '$'.

5.3. Fehlerbehandlung

Alle Fehlerzustände sind einfach erfassbar und vom Programmierer behandelbar.

Beispiel:

```
begin
  puts "open"
  f = File.open("gibtsnicht")
  puts "erfolgreich"
rescue
  puts $!
  # leere Datei erzeugen
  File.open("gibtsnicht", "w") {}
  retry # und nochmal von vorne
end
```

Wenn hier ein Fehler auftritt, wird das Programm bei "rescue" weiter abgearbeitet.

5.4. Aufzähler. und Böcke

Ein "Iterator" ist eine Methode die ein Programm-Stück mit Daten ausführt.

Beispiel 1:

```
  data = [1, 2, 3]
  data.each do |i|
    print i, "\n"
  end
```

Beispiel 2:

```
[ /No entries found/, /No match for/,
 /NO MATCH[:] This/, /^No Match/ ].each { |m|
  if ( m.match( data ) )
    STDOUT.print "not found\n"
    exit
  end
}
```

Dieses Prinzip funktioniert mit jeder Menge auch mit unterschiedlichen Objekten von unterschiedlichen Klassen.

5.5. Programm-Abschnitte

In normalen Programmiersprachen schreibt man Programme of mehrfach wenn sie mit unterschiedlichen Daten umgehen. Das ist in Ruby nicht nötig, sogar Programm-Teile lasen sich öfters anwenden, oder an Unterprogramme übergeben.

Beispiel:

```
class File
  def File.openAndProcess(*args)
    f = File.open(*args)
    yield f
    f.close()
  end
end

File.openAndProcess("testfile", "r") do |aFile|
  print while aFile.gets
end
```

5.6. Dynamische Speicherverwaltung

Um die Speicherverwaltung muss man sich nicht selbst kümmern. Das erledigt Ruby selbst und entsorgt nicht mehr benutze Objekte.

5.7. Erweiterung in C

Das Schreiben von eigenen Erweiterungen in C ist extrem einfach. Man kann die Speicherverwaltung von Ruby in C benutzen. Zum anderen kann man aus vorhandene Bibliotheken fast automatisch eine Ruby Erweiterung generieren.

5.8. und viel viel mehr

Dies ist nur eine kleine Auswahl, Im Netz oder in den Büchern sind diese Punkte ausführlicher erklärt.

6. Ruby und das Web

6.1. Ruby CGI und mod_ruby

Ruby kann gerade im Web seine Leistungsfähigkeit unter Beweis stellen. Als komplexes Beispiel hier das Senden von Dateien über den Web-Browser.

Beispiel 1, index.rhtml:

```
<%
require 'cgi' # Require the CGI library

cgi = CGI.new("html4Tr")
cgi.out {
 cgi.html {
  cgi.head {
'<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1">
<link rel="stylesheet" type="text/css" href="/style.css">
'+  cgi.title{"Upload einer Datei"} } +
  cgi.body {
   cgi.h1 { "Datei zum Upload auf den Server angeben" } +
   cgi.center {
   cgi.multipart_form("save.rhtml"){
    cgi.file_field("save", 40) +
    cgi.submit
}}}}}
%>
```

Beispiel 2, save.rhtml:

```
<%
require 'cgi' # Require the CGI library
```

```
cgi = CGI.new()  # New CGI object
upload = cgi['save']
if upload.nil?
  print " Keine Datei angegeben\r\n"
  exit
end

dir = 'upload'
check = "#{dir}/#{upload.original_filename}"
check.untaint

if FileTest::exists?( check )
  print " Datei ist bereits vorhanden, Abbruch!\r\n"
else
  File.open( check, 'w+') { |file|
    file << upload.read
  }
  print " Datei gespeichert.\r\n"
end
%>
```

6.1. Ruby auf Schienen

"Ruby on Rails" ist ein Framework, was die Programmierung von Web -Applikationen mit Datenbanken sehr vereinfacht. Ruby erobert zurzeit das Web, ein markantes Beispiel war die Migration des Forums "eins.de" von PHP auf "Ruby on Rails". Aus 50.000 Zeilen PHP wurden wurden gerade mal 5000 Zeilen Ruby.

7. Schlagworte

vollständig Objektorientiert
Prinzip der geringsten Überraschung (POLS)
Einfach zu verstehen, transparent (KISS)
Wiederhole dich nicht (DRY)
Es gibt mehr als einen Weg ein Sache zu tun.

8. Bücher

Programmieren mit Ruby
Armin Roehrl, Stefan Schmiedl, Clemens Wyss
Dpunkt Verlag

Rapid Web Development mit Ruby on Rails
Ralf Wirdemann und Thomas Baustert
Hanser Fachbuchverlag

8. Dokumente im Netz

http://www.ruby-lang.org/en/
http://www.ruby-doc.org/
http://www.rubycentral.com/book/index.html
http://www.rubycentral.com/ref/
http://www.caliban.org/ruby/rubyguide.shtml
http://www.glue.umd.edu/~billtj/ruby.html
http://docs.rubygems.org/
http://www.rubyonrails.org/
http://api.rubyonrails.org/
http://www.ruby-doc.org/stdlib/libdoc/cgi/rdoc/
http://wiki.modruby.net/en/
http://sean.chittenden.org/programming/ruby/mod_ruby/apachecon-2002/mod_ruby_intro.html
http://wxruby.rubyforge.org/wiki/wiki.pl?WxRuby_Tutorial
http://poocs.net/articles/2006/03/13/the-adventures-of-scaling-stage-1
http://www.approximity.com/rubybuch2/rb_main.html
http://www.rubyonrails-ug.de/
http://www.b-simple.de/
http://oss.erdfunkstelle.de/ruby/FAQ.html
http://www.dinoex.info/ruby.html