Ruby

A Programmer's Best Friend

Arbeiten mit Ruby - Datentypen, einfache und komplexe Datenstrukturen

Andreas Neumann M.A.

08.05.12

Ruby - alles ist ein Objekt

- alles in Ruby ist ein Objekt
- .methods gibt eine Liste aller möglicher Methoden des Objekts aus

SOAP — ruby — 118×35

ruby | ruby

```
1.9.2-p290 :011 > 1
 => 1
1.9.2-p290 :012 > 1.m
1.magnitude  1.method     1.methods    1.modulo
1.9.2-p290 :012 > 1.methods
 => [:to_s, :-, :, :-, :, :, :div, :, :modulo, :divmod, :fdiv, :, :abs, :magnitude, :, :, :=>, :, :, :, :, :, :, :, :,
 :[], :, :, :to_f, :size, :zero?, :odd?, :even?, :succ, :integer?, :upto, :downto, :times, :next, :pred, :chr, :ord, :
to_i, :to_int, :floor, :ceil, :truncate, :round, :gcd, :lcm, :gcdlcm, :numerator, :denominator, :to_r, :rationalize, :
singleton_method_added, :coerce, :i, :, :eql?, :quo, :remainder, :real?, :nonzero?, :step, :to_c, :real, :imaginary, :
imag, :abs2, :arg, :angle, :phase, :rectangular, :rect, :polar, :conjugate, :conj, :pretty_print_cycle, :pretty_print,
 :between?, :po, :poc, :pretty_print_instance_variables, :pretty_print_inspect, :nil?, :, :!, :hash, :class, :singleto
n_class, :clone, :dup, :initialize_dup, :initialize_clone, :taint, :tainted?, :untaint, :untrust, :untrusted?, :trust,
 :freeze, :frozen?, :inspect, :methods, :singleton_methods, :protected_methods, :private_methods, :public_methods, :in
stance_variables, :instance_variable_get, :instance_variable_set, :instance_variable_defined?, :instance_of?, :kind_of
?, :is_a?, :tap, :send, :public_send, :respond_to?, :respond_to_missing?, :extend, :display, :method, :public_method,
:define_singleton_method, :__id__, :object_id, :to_enum, :enum_for, :pretty_inspect, :ri, :equal?, :!, :!, :instance_e
val, :instance_exec, :__send__]
1.9.2-p290 :013 > "Hallo Welt"
 => "Hallo Welt"
1.9.2-p290 :014 > true
 => true
1.9.2-p290 :015 > false
 => false
1.9.2-p290 :016 > nil
 => nil
1.9.2-p290 :017 > nil.methods
 => [:to_i, :to_f, :to_s, :to_a, :inspect, :, :, :, :nil?, :to_r, :rationalize, :to_c, :pretty_print_cycle, :po, :poc,
 :pretty_print, :pretty_print_instance_variables, :pretty_print_inspect, :, :, :!, :eql?, :hash, :=>, :class, :singlet
on_class, :clone, :dup, :initialize_dup, :initialize_clone, :taint, :tainted?, :untaint, :untrust, :untrusted?, :trust
, :freeze, :frozen?, :methods, :singleton_methods, :protected_methods, :private_methods, :public_methods, :instance_va
riables, :instance_variable_get, :instance_variable_set, :instance_variable_defined?, :instance_of?, :kind_of?, :is_a?
, :tap, :send, :public_send, :respond_to?, :respond_to_missing?, :extend, :display, :method, :public_method, :define_s
ingleton_method, :__id__, :object_id, :to_enum, :enum_for, :pretty_inspect, :ri, :, :equal?, :!, :!, :instance_eval, :
instance_exec, :__send__]
1.9.2-p290 :018 >
```

Ruby einfache Literale

- Ruby kennt unter anderem
- Integers
- Doubles
- Strings
- Sybols
- true und false
- nil

```
 => 1
1.9.2-p290 :054 > 1.6
 => 1.6
1.9.2-p290 :055 > "Ein Test"
 => "Ein Test"
1.9.2-p290 :056 > :key
 => :key
1.9.2-p290 :057 > nil
 => nil
1.9.2-p290 :058 >
```

Ruby - Collections / Datenstrukturen

- Die am häufigsten verwendeten Datenstrukturen in Ruby sind Arrays und Hashes

Ruby Array

- [] oder Array.new
- array[index] erlaubt es auf bestimmte Positionen zuzugreifen
- ist veränderbar und hat keine vorgegebene Größe

```
> a_array = [1,2,-"drei",:vi-er]
=> [1, 2, "drei", :vier]

> a_array.each do |wert|
.. puts wert
.. end
=> "12dreivier"

> a_array[0]
=> 1

> a_array[0] = "null"
=> "null"

> a_array
=> ["null", 2, "drei", :vier]

> a_array << 5
=> ["null", 2, "drei", :vier, 5]
```

Ruby Map / Hash

- {} oder Hash.new
- key => value
- ab 1.9 auch key : value
- veränderbar
- zugriff über key
- bei nicht vorhandenem key nil

```
> hash = {"eins" => 1, "zwei" => 2}
=> {"eins"=>1, "zwei"=>2}
> hash["eins"]
=> 1
> hash["drei"] = 3
=> 3
> hash
=> {"eins"=>1, "zwei"=>2, "drei"=>3}
> hash.keys
=> ["eins", "zwei", "drei"]
> hash.values
=> [1, 2, 3]
> hash.size
=> 3
> hash["vier"]
=> nil
```

Komplexe Datenstrukturen

- Komplexe Datenstrukturen lassen sich durch v,erschachtelung einfacher Datenstrukturen wie Arrays und Hashes erreichen

```
> matrix = [[1,2],[3,4,5]]
=> [[1, 2], [3, 4, 5]]
> matrix[1]
=> [3, 4, 5]
> matrix[0][1]
=> 2
> matrix[1][1]
=> 4
> matrix[1][6]
=> nil
```

```
> complex = {"tage" => ["Montag", "Dienstag", "Mittwoch"], "monate" => ["Januar"]}
=> {"tage"=>["Montag", "Dienstag", "Mittwoch"], "monate"=>["Januar"]}
> complex["tage"]
=> ["Montag", "Dienstag", "Mittwoch"]
> complex["tage"][1]
=> "Dienstag"
```