Inspektion zur Laufzeit

Metaprogrammierung

HPI Hasso Plattner Institut

Ruby als interpretierte Sprache

- Alles ist ein Objekt (auch Klassen)
- Monkey-Patching

Rubys Klassen und Objekte sind zur Laufzeit des Programms anpassbar!

Arten der Metaprogrammierung

- Inspektion
  - Klassen und Objekte untersuchen
  - Verfügbare Methoden, Klassenhierarchie inspizieren
- Manipulation
  - Objekte erweitern
  - Klassen generieren

Object#inspect

```ruby
42.inspect
# => "42"

"42".inspect
# => "\"42\""

Greeter.new.inspect
# => "#<Greeter:0x0000562cd7b50040>"
```

`Object#inspect` gibt eine menschenlesbare Repräsentation des Objekts als Zeichenkette zurück.

Module#instance_methods

```
Object.instance_methods
# => [:instance_of?, :public_send, :instance_variable_get,
:instance_variable_set, :instance_variable_defined?,
:remove_instance_variable, :private_methods, :kind_of?,
:instance_variables, :tap, :is_a?, :extend, :to_enum,
:enum_for, :pp, :<=>, :===, :=~, :!~, :eql?, :respond_to?,
:freeze, :inspect, :display, :object_id, :send, :to_s,
:method, :public_method, :singleton_method,
:define_singleton_method, :nil?, :hash, :class,
:singleton_class, :clone, :dup, :itself, :yield_self, :taint,
:tainted?, :untaint, :untrust, :trust, :untrusted?, :methods,
:protected_methods, :frozen?, :public_methods,
:singleton_methods, :!, :==, :!=, :__send__, :equal?,
:instance_eval, :instance_exec, :__id__]

Object.new.methods
# => [:instance_of?, ...]
```

Die Klasse `Method`

```ruby
[].method(:map)
# => #<Method: Array#map>

[].method(:reduce).owner # => Enumerable
[].method(:map).owner    # => Array
```

`Method#owner` – Gibt Definitionsort (Modul oder Klasse) zurück

`Method#arity` – Gibt die Anzahl der Parameter zurück

`Method#call` – Ruft die Methode auf

Methoden als Blöcke verwenden

```ruby
def add_two(element)
  element + 2
end

[1, 2, 3].map &method(:add_two)
# => [3, 4, 5]
```

Module#ancestors

- Gibt alle Superklassen und -module (!) zurück
- Relevant für Methodenaufrufe und `super`
  - Jedes Element dieser Liste nach Methode fragen

```
Array.ancestors
# => [Array, Enumerable, Object, Kernel, BasicObject]

require "pp"

Array.ancestors
# => [Array, Enumerable, Object, PP::ObjectMixin, Kernel,
BasicObject]
```

Inspektion zur Laufzeit