

# Metaprogrammierung - weniger ist mehr

Vladimir Dobriakov, innoQ Deutschland GmbH

2. September 2009, Rails-Konferenz, Offenbach/Main

# Ziel

Zielumgebung

# Ziel

## Zielumgebung

- JRuby?
- MRI (Matz's Ruby Implementierung)?

# Ziel

## Zielumgebung

- JRuby?
- MRI (Matz's Ruby Implementierung)?

## Zielgruppe

# Ziel

## Zielumgebung

- JRuby?
- MRI (Matz's Ruby Implementierung)?

## Zielgruppe

- Deine Kollegen?

# Ziel

## Zielumgebung

- JRuby?
- MRI (Matz's Ruby Implementierung)?

## Zielgruppe

- Deine Kollegen?
- Du selbst in zwei Wochen?

# Programming by intention

## Warum Ruby?

# Programming by intention

## Warum Ruby?

- weil Ruby so schnell ist?

# Programming by intention

## Warum Ruby?

- weil Ruby so ~~schnell~~ ist?

# Programming by intention

```
Order.pending.each do |order|  
  order.ship! unless  
    order.customer.in_debt?  
end
```

## Warum Ruby?

- weil Ruby so ~~schnell~~ ist?
- lockere Syntax

# Programming by intention

```
Order.pending.each do |order|  
  order.ship! unless  
    order.customer.in_debt?  
end
```

## Warum Ruby?

- weil Ruby so ~~schnell~~ ist?
- lockere Syntax
- unless / if modifiers

# Programming by intention

```
Order.pending.each do |order|  
  order.ship! unless  
    order.customer.in_debt?  
end
```

## Warum Ruby?

- weil Ruby so ~~schnell~~ ist?
- lockere Syntax
- unless / if modifiers
- Synonyme wie length / size

# Programming by intention

## Programming by intention

# Metaprogrammierung

- `define_method`
- Methoden definieren mit `eval`, `class_eval` or `module_eval`
- `const_missing`
- `Class.new`
- Methoden nach Bedarf mit `method_missing`
- Aliasing and chaining methods
- Monkey patching
- Hooks (`inherited`, `included`, `method_added`)
- Delegation

# Statische Definitionen

- statische Definition

```
class Foo
  def bar; end
end
```

- Klassen wieder öffnen, methoden umdefinieren

```
class Foo
  def bar
    puts "redefined"
  end
end
```

- alias :size, :length

# eval

```
module Storage
  DESIRED_METHODS.each do |method|
    eval <<EOF
      def #{method}
        Thread.current[:fast_gettext_#{method}]
      end
    EOF
  end
end
```

`alias :evil, :eval`

## Problem mit Stack Trace

```
21 def translate(key)
=> 22   found = FastGettext.current_cache[...]
    23   ...
    24 end

(rdb:4) s
[-3, 6] in (eval)
*** No sourcefile available for (eval)
(eval):2
```

## Kleine Abhilfe

```
%w(edit new).each do |action|
  module_eval <<-EOT, __FILE__, __LINE__
  def #{action}_polymorphic_url(record_or_hash, options = {})
    polymorphic_url(
      record_or_hash,
      options.merge(:action => "#{action}"))
  end

  def #{action}_polymorphic_path(record_or_hash, options = {})
    polymorphic_url(
      record_or_hash,
      options.merge(:action => "#{action}", :routing_type => :path))
  end
  EOT
end
```

## module\_eval + define\_method

Mehr eval:

- class\_eval
- module\_eval

Akzeptiert:

- string
- block

```
Storage.module_eval do
  define_method method_name do
    ...
  end
end
```

## Closure mit `define_method`

```
key = "fast_gettext_#{method_name}"  
Storage.define_method method_name do  
  Thread.current[key]  
end
```

# Vergleich

	Closure	Parametrierung	Performance	leichte Wartung
<code>eval</code>	nein	● (nur Strings)	●●	○
<code>class_eval (string)</code>	nein	● (nur Strings)	●●	○
<code>define_method</code>	ja	●●	●	●
<code>def</code>	nein	○	●●	●●

Legende: ●● am Besten, ● OK, ○ problematisch

# Fortgeschrittene Beispiele



# Anwendungsspezifische Exceptions

```
class TransactionAbortedError < StandardError; end  
class TransactionCommitNotAllowed < StandardError; end  
class Disconnect < StandardError; end
```

## Wiederholung vermeiden

```
exceptions = %w( TransactionAborted  
  TransactionCommitNotAllowed Disconnect )  
  
exceptions.each { |e|  
  const_set(e, Class.new(StandardError))  
}
```

## Nach Bedarf erzeugen

```
class SmarterThing
  def self.const_missing(error_name) #:nodoc:
    if error_name.to_s =~ /Error\z/
      const_set(error_name,
                Class.new(StandardError))
    else
      super
    end
  end
end
```

```
raise SmarterThing.DisconnectError
```



**DRY = single point of truth**

**DRY != lzw compression**

## Besserer Fall für Class.new - isoliertes Testen

```
test 'initial state' do
  c = Class.new do
    include Workflow
  end
  # assert something
  c.class_eval do
    workflow { state :one; state :two }
  end
  # assert something else
end
```

# Neue Klassen am laufenden Band



# Struct.new

```
RolePermission = Struct.new(:rolename, :roletitle, :domain) do
  def html_id(prefix, suffix)
    "#{prefix}_#{self.rolename}_#{self.domain}_#{suffix}"
  end
end
```

```
ZipCode.all.each do |zip|
  res << RolePermission.new(role.rolename, '...', zip)
end
```

```
<label for="<%= permission.html_id('checkbox', user.id) %>"
  class="checkbox_roles"><%= permission.roletitle %>
</label>
```



## acts\_as\_state\_machine und workflow

```
class Order < ActiveRecord::Base
  include Workflow
  workflow do
    state :submitted do
      event :accept, :transitions_to => :accepted do
        |reviewer, args|
          puts "accepted by #{reviewer}"
        end
      end
    end
    state :accepted do
      event :ship, :transitions_to => :shipped
    end
    state :shipped
  end
end
```

## workflow - Benutzung

```
o = Order.find(123)
o.ship!
assert o.shipped?
```

## workflow - Originalimplementierung

```
def patch_context(context)
  context.instance_variable_set("@workflow", self)
  context.instance_eval do
    alias :method_missing_before_workflow :method_missing
    def method_missing(method, *args)
      if potential_methods.include?(method.to_sym)
        @workflow.send(method, *args)
      else
        method_missing_before_workflow(method, *args)
      end
    end
  end
end
```

# `method_missing` NACHTEILE

## method\_missing - direkte Probleme

- Introspection

```
order.public_methods
```

## method\_missing - direkte Probleme

- Introspection

```
order.public_methods
```

- Dokumentation

## method\_missing - direkte Probleme

- Introspection

```
order.public_methods
```

- Dokumentation
- code completion

# method\_missing Implementierungsprobleme

- eine sehr lange Methode für unterschiedliche Aspekte

```
if /^find_(all_by|by)_([\_a-zA-Z]\w*)$/.  
  match(method_id.to_s)  
  # do one thing  
elsif /^find_or_(initialize|create)_by_(\[_a-z.../  
  # do another thing  
else  
  super  
end
```

# method\_missing Implementierungsprobleme

- eine sehr lange Methode für unterschiedliche Aspekte

```
if /^find_(all_by|by)_([\_a-zA-Z]\w*)$/.  
  match(method_id.to_s)  
  # do one thing  
elsif /^find_or_(initialize|create)_by_(\[_a-z.../  
  # do another thing  
else  
  super  
end
```

- langsamer Lookup

# method\_missing Implementierungsprobleme

- eine sehr lange Methode für unterschiedliche Aspekte

```
if /^find_(all_by|by)_([\_a-zA-Z]\w*)$/.  
  match(method_id.to_s)  
  # do one thing  
elsif /^find_or_(initialize|create)_by_(\[_a-z.../  
  # do another thing  
else  
  super  
end
```

- ~~langsamer Lookup~~

## `method_missing` Rails Tricks

- `generate_read_methods` von ActiveRecord
- Rails Routing

## workflow - Originalimplementierung

```
def patch_context(context)
  context.instance_variable_set("@workflow", self)
  context.instance_eval do
    alias :method_missing_before_workflow :method_missing
    def method_missing(method, *args)
      if potential_methods.include?(method.to_sym)
        @workflow.send(method, *args)
      else
        method_missing_before_workflow(method, *args)
      end
    end
  end
end
```

# Besserer Weg

`define_method`

statt

`method_missing`

## define\_method

```
def workflow(&specification)
  @workflow_spec = Specification.new(Hash.new, &specification)
  @workflow_spec.states.values.each do |state|
    state_name = state.name
    module_eval do
      define_method "#{state_name}?" do
        state_name == current_state.name
      end
    end
  end
end

...
```

# The Rails way

- ein Modul für Instanzmethoden
- ein anderes Modul für Klassenmethoden
- benutze 'included' Hook
- include (Mixin) zum Einbetten in die Klasse

# Methoden ablegen

```
module WorkflowClassMethods
  def workflow(&specification)
end

module WorkflowInstanceMethods
  def current_state
end

module ActiveRecordInstanceMethods
  def load_workflow_state
  def persist_workflow_state(new_value)
end
```

## 'included' Hook

```
module Workflow
  def self.included(klass)
    klass.send :include, WorkflowInstanceMethods
    klass.extend WorkflowClassMethods
    if Object.const_defined?(:ActiveRecord)
      if klass < ActiveRecord::Base
        klass.send :include, ActiveRecordInstanceMethods
        klass.before_validation :write_initial_state
      end
    end
  end
end
```

## Benutzung (mixin)

```
class Article
  include Workflow
  workflow do
    state :awaiting_review do
      event :review, :transitions_to => :being_reviewed
    end
    state :being_reviewed
  end
end
```

```
article = Article.new
article.review!
article.being_reviewed?
```

# Gutes method\_missing

Domain specific languages (DSL):

- rake
- XML builder

```
xml = Builder::XmlMarkup.new
xml.user :id => @user.id do |u| # <user id="123">
  u.email @user.email # <email>joe...</email>
  u.subscriptions do |s|
    @user.subscriptions do |subscr|
      s.topic :href => ...
    end
  end
end
```

# method\_missing

```
1) Error:  
NoMethodError: undefined method  
'user_my_widgets_nil_class_url' for #
```

## method\_missing

```
1) Error:  
NoMethodError: undefined method  
'user_my_widgets_nil_class_url' for #
```

**Baue gute Fehlermeldungen!**

# Opinionated software



## active\_record/validations.rb

```
def full_messages
  full_messages = []
  @errors.each_key do |attr|
    @errors[attr].each do |msg|
      next if msg.nil?
      if attr == "base"
        full_messages << msg
      else
        full_messages << @base.class.
          human_attribute_name(attr) + " " + msg
      end
    end
  end
end
full_messages
```

## gettext

```
module ActiveRecord
  class Errors
    def full_messages_with_gettext
      full_messages = []
      errors = localize_error_messages
      errors.each_key do |attr|
        errors[attr].each do |msg|
          next if msg.nil?
          full_messages << msg
        end
      end
      full_messages
    end
    alias_method_chain :full_messages, :gettext
  end
end
```

## alias\_method\_chain

```
module ActiveRecord
  class Errors
    def localize_with_better_default(append_field = false)
      localize_without_better_default(append_field)
    end
    alias_method_chain :localize, :better_default
  end
end
```

# Monkey patching Problem

- geht kaputt bei jedem Rails upgrade
- als letzter Ausweg
- schreibt eine Liste der monkey patches auf! - das sind Todos/Warnungen
- legt in Rails Initializers ab!
- vielleicht als Wrapper oder Vererbung implementieren

# Entwurfsmuster/Techniken

- Intention zum Ausdruck bringen statt nur Wiederholung vermeiden

# Entwurfsmuster/Techniken

- Intention zum Ausdruck bringen statt nur Wiederholung vermeiden
- anonyme Class.new fürs Testen

## Entwurfsmuster/Techniken

- Intention zum Ausdruck bringen statt nur Wiederholung vermeiden
- anonyme `Class.new` fürs Testen
- Wrapper, Vererbung statt Patches

## Entwurfsmuster/Techniken

- Intention zum Ausdruck bringen statt nur Wiederholung vermeiden
- anonyme `Class.new` fürs Testen
- Wrapper, Vererbung statt Patches
- `define_method` statt `method_missing` benutzen wann immer möglich

# Entwurfsmuster/Techniken

- Intention zum Ausdruck bringen statt nur Wiederholung vermeiden
- anonyme `Class.new` fürs Testen
- Wrapper, Vererbung statt Patches
- `define_method` statt `method_missing` benutzen wann immer möglich
- monkey patching (als letzter Ausweg), benutze Rails Initializers, "technische Schulden"

Bringt eure Ziele und die Lösung  
zum Ausdruck  
mit

Ruby!

Setzt Metaprogrammierung

**weise**

**ein!**

## Contact



innoQ Deutschland GmbH  
Halskestrasse 17  
D-40880 Ratingen  
Tel +49 2102 77 162-100  
Fax +49 2102 77 1601

**Vladimir Dobriakov**  
<http://blog.geekQ.net>

<http://www.innoq.com>  
[info@innoq.com](mailto:info@innoq.com)

innoQ Schweiz GmbH  
Gewerbestrasse 11  
CH-6330 Cham  
Tel +41 41 743 01 11

# Resources

- <http://www.flickr.com/photos/pinksherbet/3484925590/>  
by "D Sharon Pruitt"
- <http://www.flickr.com/photos/endlisnis/458855308/>
- <http://www.flickr.com/photos/randomliteraturecouncil/1812738660/>
- <http://www.flickr.com/photos/dannyboyster/60371673/>