Rails Workshop #4

Modularisierung

Abstract

Ruby bietet viel Möglichkeiten
um Code gut zu **strukturieren**.

Rails bietet viele Möglichkeiten
um Code zu **organisieren**.

Sehen wir uns an welche
Möglichkeiten es gibt!

Vererbung

Wie auch in allen anderen OOP Sprachen gibt es in Ruby Vererbung.

class basics

```
class Andy
  def me
    "Andy"
  end

  def self.you
    "Betty"
  end
end
> Andy.you => "Betty"
> Andy.me => NoMethodError: undefined method `me' for Andy:Class
> Andy.new.me => "Andy"
```

module basics

```
module Kids
  def small_kids
    ['Kiana', 'Naya']
  end

  def self.big_kids
    ['Slash', 'Ozzy']
  end
end

> Kids.small_kids => NoMethodError: undefined method `small_kids' for Kids:Module
> Kids.big_kids => ["Slash", "Ozzy"]
```

class include module basics

```
class Kindergarden
  include Kids

  def very_small_kids
    small_kids
  end
end

> Kindergarden.very_small_kids
NoMethodError: undefined method `very_small_kids' for Kindergarden:
Class
> Kindergarden.new.very_small_kids => ["Kiana", "Naya"]
```

Intermezzo: attr_reader, attr_writer, attr_accessor

`@name eine Instanz Variable`	`def initialize(name)`
	`  @name = name`
	`end`

`attr_reader :name`	`def name`
	`  @name`
	`end`

`attr_writer :name`	`def name=(name)`
	`  @name = name`
	`end`

`attr_accessor :name`	`beide`

http://stackoverflow.com/questions/5046831/why-use-rubys-attr-accessor-attr-reader-and-attr-writer

Bsp: Player

```
class Player

  attr_reader :first_name, :last_name

  def initialize(options = {})
    @first_name = options.fetch(:first_name, 'Harry')
    @last_name  = options.fetch(:last_name, 'Belafonte')
  end

  def name
    "#{first_name} #{last_name}"
  end
end
```

Bsp.: PlayerCar

```
class PlayerCar < Player

  attr_reader :car

  def initialize(options = {})
    super
    @car = options.fetch(:car, 'Porsche')
  end

  def car_info
    "#{name} drives a #{car}"
  end
end
```

Call it baby:

```
 :001 > p = PlayerCar.new
 => #<PlayerCar:0x007ff27b45ced0 @first_name="Harry", @last_name="
Belafonte", @car="Porsche">
 :002 > p.car_info
 => "Harry Belafonte drives a  Porsche"
```

Composing with Mixins

code example

instance_eval vs. class_eval

instance_eval -> class

class_eval -> instance

WTF?

http://stackoverflow.com/questions/4409023/rubys-def-and-instance-eval-vs-class-eval

Code example

instance_eval auch für module

```
:018 > Family::Parent.instance_eval do
:019 >     def fish
:020?>       "Wanda"
:021?>     end
:022?>   end
=> :fish
:023 > Family::Parent.fish
=> "Wanda"
```

class_eval auch für module

```
:032 >   Family::Parent.class_eval do
:033 >     def self.dog
:034?>       "Achmed"
:035?>     end
:036?>   end
=> :dog
:037 > Family::Parent.dog
=> "Achmed"
```

Concerns

Concerns are pieces of code that allow you to better organize the code that you write.

Controller, Model

Code example

gutes Beispiel in ACE: app/models/concerns/couch_helper.rb

guter Post: http://richonrails.com/articles/rails-4-code-concerns-in-active-record-models

Decorators

Draper: View Models for Rails

https://github.com/drapergem/draper

Was ist Draper?

“Draper adds an object-oriented layer of presentation logic to your Rails application.

Without Draper, this functionality might have been tangled up in procedural helpers or adding bulk to your models. With Draper decorators, you can wrap your models with presentation-related logic to organise - and test - this layer of your app much more effectively.”

app vs. lib

Wann gehört ein *Model* oder eine *Class* in das **app** Verzeichnis und wann in **lib**?

app

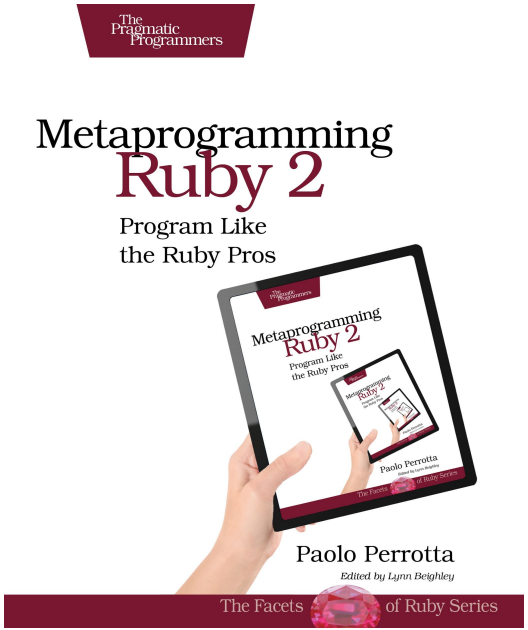
*Wenn ein model Zugriff auf die Datenbank hat oder Business-Logik genau für diese Applikation beinhaltet, gehört es ausschließlich zu dieser Applikation und wird in **app** palziert.*

lib

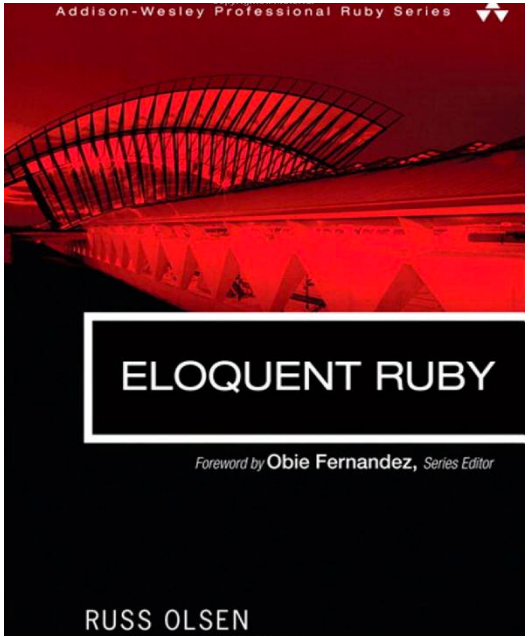
Wenn ein model keinen direkten Datenbankzugriff hat und Code beinhaltet, der auch extrahiert werden könnte um in einer anderen Applikation zu nutzen, gehört es nicht ausschließlich zu dieser Applikation und wird in ***lib*** *plaziert*

Fragen?

Bücher



https://pragprog.com/book/ppmetr2/metaprogramming-ruby-2



http://eloquentruby.com/

DANKE :)