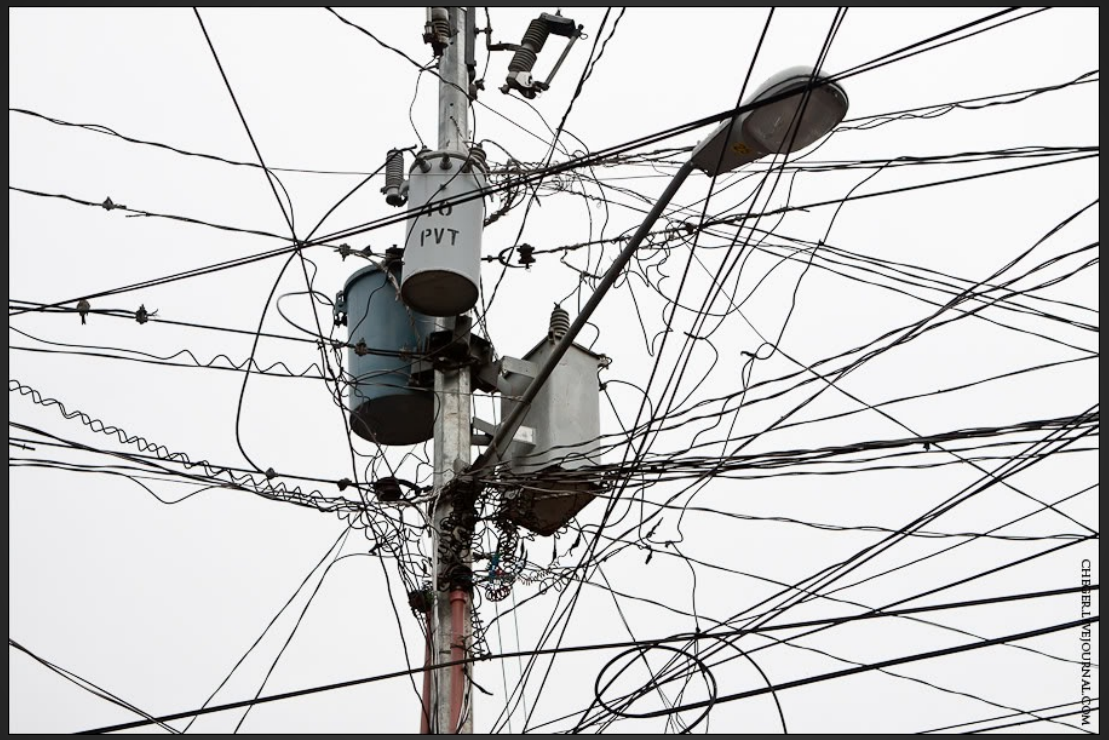
Traits

Существительное.

Особенности, признак, характерные черты

англо-русский универсальный дополнительный переводческий словарь И. Моститского

ООП



...в том числе предполагалось легкое переиспользование кода.

Оказалось, не так-то просто

Слишком различающаяся логика, слишком разные бизнес-правила

Тогда придумали Traits

- Механизм повторного использования кода
- Похожи на mixins (модули), да не они
- Могут быть параметризируемыми
- Конфликтов имен не должно быть - Traits должен обеспечивать данный механизм разрешения конфликтов
- Traits могут требовать от включающего их класса доступ к методам
- Traits не общаются с shared переменными инстанса

http://scg.unibe.ch/archive/papers/Scha03a**Traits**.**pdf**

Как выглядит (scala):

```
abstract class Spacecraft{
    def engage
}

trait PulseEngine{
    val maxPulse: Int
    var currentPulse = 0;
    def speedUp { if( currentPulse < maxPulse ) currentPulse += 1  }
}


trait ControlCabin{
    def engage = increaseSpeed
    def increaseSpeed
}

class StarCruiser extends Spacecraft with CommandoBridge with PulseEngine{
    val maxPulse = 200
}
```

Что насчет руби?

```
module HtmlLinkTrait
    def cleanup_link_field(link)
        .....
    end
end

class MyModel
    include HtmlLinkTrait

    def before_save
        cleanup_link_field(link)
    end
end
```

```
class UserProfile
    include HtmlLinkTrait

    def before_save
        cleanup_link_field(link)
    end
end
```

Круто. Но не очень.

...тогда придумали ActiveSupport::Concern

Используем его!

```
module HtmlLinkTrait
    include ActiveSupport::Concern

    included do
        before_save :cleanup_link_field

    end

    module ClassMethods
        def  find_broken_links
        end
    end

    def cleanup_link_field(link = self.link)
        .....
    end
end
```

```
class MyModel
    include HtmlLinkTrait
end

class UserProfile
    include HtmlLinkTrait
end
```

Неудобно, что у любой модели, которая хочет использовать наш HtmlLinkTrait должно быть поле link

А вдруг, в другой модели уже есть site_link ?

Опять код нельзя переиспользовать?

... И тогда я решил сделать свою библиотеку поддержки traits с блекджеком и шлюхами

... ведь её никто за меня не сделает!

gem penetrator

https://github.com/svenyurgensson/penetrator

- Повторяет функционал ActiveSupport::Concern
- Вводит метод класса:
  - behaves_like 'my_htmlizer', arg1, arg2, ... argN
- То есть, позволяет задавать параметры для импортируемого модуля-traits!
- И этот traits может использовать разные методы для разных классов в зависимости от аргументов

```ruby
# app/traits/can_have_args_trait.rb
module CanHaveArgsTrait
      extend Penetrator::Concern
      included do |*args|
        args.each do |method_name|
          define_method(method_name) do
            method_name.to_s + "-chunked!"
          end
        end
      end # included
end # CanHaveArgs


# app/models/my_model.rb
class Victim
     behaves_like :CanHaveArgs, 'arg1', 'arg2'
end

obj = Victim.new
obj.arg1  # => 'arg1-chunked!'
obj.arg2  # => 'arg2-chunked!'
```

```
module Outer
  module InnerTrait
      include Penetrator::Concern
      def inner
            ....
      end
   end
end


class Victim
      behaves_like 'outer/inner' # or behaves_like :Outer_Inner
end
```

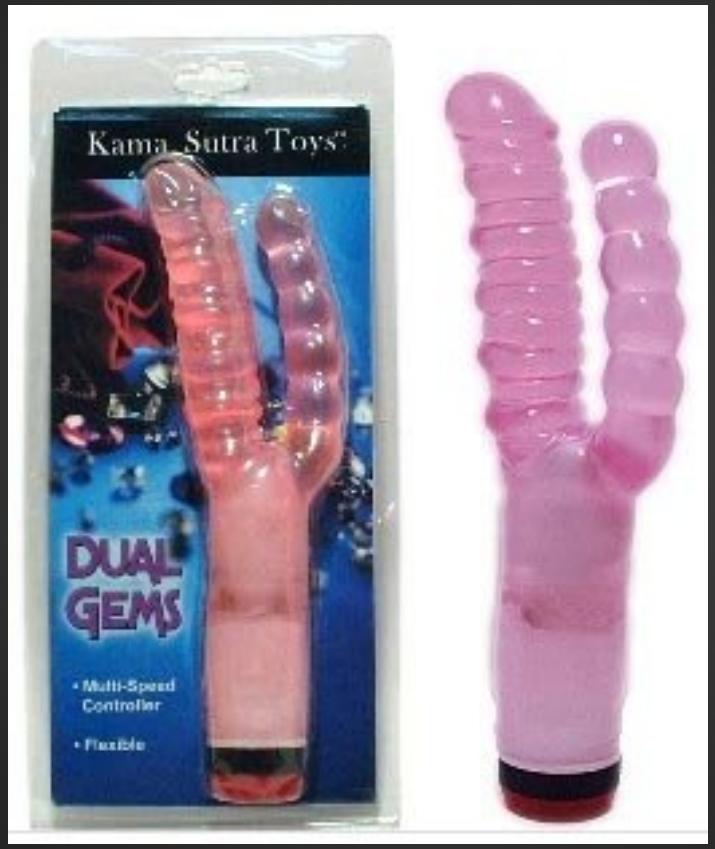
Найдет и подключит то, что надо

Для чего?

- Многие модели используют seo_optimizer_trait
- Поля модели разные, но код трейтса - один для всех
- Многие модели используют html_link_normalizer_trait
- Многие модели используют sluggable_trait
- Большинство контроллеров используют crudable_trait пример:

```
class BlogsController < FrontendController

  private

  def resource_class
    BlogPost
  end

  behaves_like "crudable"

  private
  def take_layout
    'two_column'
  end

  def default_order
    "created_at desc"
  end

end
```

Looks forward! Soon....



Double Penetrator™ !!!