Krotko o callbackach w RoR

```ruby
class User < ActiveRecord::Base
  before_save :login_required

  def login_required
    return nil
  end
end
```

```ruby
  class User < ActiveRecord::Base
    before_save :login_required

    def login_required
      return nil
    end
  end

User.new.save #=> ?
```

```
  class User < ActiveRecord::Base
    before_save :login_required

    def login_required
      return nil
    end
  end

User.new.save #=> true
```

```ruby
class UserController < ApplicationController
  before_filter :load_user, :load_profile

  def index
    render :text => "DRUG!"
  end

  def load_user
    return false
  end

  def load_profile
    puts "Load profile"
  end
end
```

```ruby
class UserController < ApplicationController
  before_filter :load_user, :load_profile

  def index
    render :text => "DRUG!"
  end

  def load_user
    return false
  end

  def load_profile
    puts "Load profile"
  end
end

# Czy wykona sie load_profile?
```

```ruby
class UserController < ApplicationController
  before_filter :load_user, :load_profile

  def index
    render :text => "DRUG!"
  end

  def load_user
    return false
  end

  def load_profile
    puts "Load profile"
  end
end

# Czy wykona sie load_profile?
# TAK!
```

Stworzymy czysta klase i uzyjemy ActiveSupport::Callbacks

```ruby
class User
  include ActiveSupport::Callbacks
  define_callback :save

  set_callback :save, :before, :saving_message

  def saving_message
    puts "saving..."
  end

  def save
    run_callbacks :save do
      puts "- save"
    end
  end
end
```

Co zbudowac ActiveSupport::Callbaks ?

```ruby
value = nil
halted = false

unless halted
  result = saving_message
  halted = (false)
end

value = yield if block_given? && !halted
halted ? false : (block_given? ? value : true)
```

```ruby
class User
  include ActiveSupport::Callbacks
  define_callback :save

  set_callback :save, :before, :saving_message
  set_callback :save, :before, :saving_message2

  def saving_message; puts "saving...";end
  def saving_message2; puts "saving2...";end

  def save
    run_callbacks :save do
      puts "- save"
    end
  end
end
```

```ruby
value = nil
halted = false

unless halted
  result = saving_message
  halted = (false)
end

unless halted
  result = saving_message2
  halted = (false)
end

value = yield if block_given? && !halted
halted ? false : (block_given? ? value : true)
```

```ruby
class User
  include ActiveSupport::Callbacks
  define_callback :save

  set_callback :save, :before, :saving_message
  set_callback :save, :before, :saving_message2,
                               :prepend => true

  def saving_message; puts "saving...";end
  def saving_message2; puts "saving2...";end

  def save
    run_callbacks :save do
      puts "- save"
    end
  end
end
```

```ruby
value = nil
halted = false

unless halted
  result = saving_message2
  halted = (false)
end

unless halted
  result = saving_message
  halted = (false)
end

value = yield if block_given? && !halted
halted ? false : (block_given? ? value : true)
```

```ruby
class User
  include ActiveSupport::Callbacks
  define_callback :save,
                 :terminator => "result == false"

  set_callback :save, :before, :saving_message
  set_callback :save, :before, :saving_message2,
                                   :prepend => true

  def saving_message; puts "saving...";end
  def saving_message2; puts "saving2...";end

  def save
    run_callbacks :save do
      puts "- save"
    end
  end
end
```

```ruby
value = nil
halted = false

unless halted
  result = saving_message
  # nil == false #=> false
  halted = (result == false)
end

unless halted
  result = saving_message2
  halted = (result == false)
end

value = yield if block_given? && !halted
halted ? false : (block_given? ? value : true)
```

```
  class UserController < ApplicationController
    before_filter :load_user, :load_profile

    def index
      render :text => "DRUG!"
    end

    def load_user
      render :text => "Helou DRUG!"
    end

    def load_profile
      puts "Load profile"
    end
  end

# Czy callback sie trzeba zatrzyma?
```

```
    class UserController < ApplicationController
      before_filter :load_user, :load_profile

      def index
        render :text => "DRUG!"
      end

      def load_user
        render :text => "Helou DRUG!"
      end

      def load_profile
        puts "Load profile"
      end
    end

# Czy callback sie trzeba zatrzyma?
# TAK! Bo... :terminator => "response_body"
```

```ruby
value = nil
halted = false

unless halted
  result = load_user
  halted = (responde_body)
end

unless halted
  result = load_profile
  halted = (responde_body)
end

value = yield if block_given? && !halted
halted ? false : (block_given? ? value : true)
```

```ruby
class User < ActiveRecord::Base
  after_save :drug1
  after_save :drug2
end

# User.new.save
# => drug1
# => drug2
```

```ruby
class UsersController < ApplicationController
  after_filter :drug1
  after_filter :drug2
end

# drug2
# drug1
```

Jak nie potrafisz zatrzymac callback'a? To co?

```ruby
def callback_method
  exec('sudo shutdown -h now')
end
```

```ruby
class Proc
  def call
    # super empty method ;-)
  end
end
```

```ruby
def callback_method
  remove_const ActiveRecord
  remove_const ActiveSupport
end
```

```
def callback_method
  class Rack
    def call(env)
      # haha !
    end
  end
end
```

```ruby
require 'file'

def callback_method
  File.rm(File.expand_path(__FILE__))
end
```

Q & A