# Ruby on Rails

Dynamic Methods

# Overview

- Definir metodos dinámicamente

# Dynamic methods

- No solamente se pueden invocar a los métodos dinámicamente sino que se pueden definir los métodos dinámicamente.
- **`define_method :method_name`** y un bloque que contiene la definición del método

# Ejemplo

```ruby
class User

  ACTIVE = 0
  INACTIVE = 1
  PENDING = 2

  attr_accessor :status

  def active?
    status == ACTIVE
  end

  def inactive?
    status == User::INACTIVE
  end

  def pending?
    status == User::PENDING
  end

end

user = User.new
user.status = 1

puts user.inactive?
#=> true
puts user.active?
#=> false
```

```ruby
class User

  ACTIVE = 0
  INACTIVE = 1
  PENDING = 2

  attr_accessor :status

  [:active, :inactive, :pending].each do |method|
    define_method "#{method}?" do
      status == User.const_get(method.upcase)
    end
  end

end

user = User.new
user.status = 1

puts user.inactive?
#=> true
puts user.active?
#=> false
```

# Ventajas

- No hay duplicación de código.
- Si nuestros estados estuviesen en una base de datos, y alguien agregara estos estados, y adaptaramos nuestro código, cada vez que se agregue un estado no habría ninguna necesidad de modificar el código.

# Entonces...

- Definir métodos dinámicamente puede reducir dramáticamente la cantidad de código.