Introducción a Ruby on Rails

Flujo de Control

Review

- Flujo de control
  - if / elsif / else
  - case
  - until / unless?
  - while / for
- Qué es true qué es false?
- Qué es y para qué sirve === ?

Flujo de Control

- if, unless, elsif, else
- No se necesita paréntesis
- Se utiliza **end** para finalizar un flujo de control

```
a = 3

if a==3
    puts "a es 3"
elsif a==5
    puts "a es 5"
else
    puts "a no es ni 3 ni 5"
end
```

```
unless a==6
    puts "a no es 6"
end
```

Flujo de Control

- while, until

```
while a > 2
    puts "a = #{a}, es mayor a 2 todavía"
    a -= 1
end

until a>=10
    puts "a = #{a}, aún no es mayor a 10"
    a +=1
end
```

Flujo de Control - Forma Inline

- if, unless, while, until – en una sola línea

```ruby
#if inline
puts "a es 5 y b es 0" if a == 5 and b == 0


#While inline
times_2 = 2
times_2 *= 2 while times_2 < 100

puts times_2 # => 128
```

True/False

- **false** y **nil** objects are false

```
puts "0 es true" if 0 # => 0 es true
puts "false es true?" if "false" # => false es true?
puts "no way - false es false" if false # => NOTHING PRINTED
puts "cadena vacía es true" if "" # => cadena vacía es true
puts "nil es true?" if "nil" # => nil es true?
puts "No! nil es false" if nil # => NOTHING PRINTED
```

Triple Equal

- Triple Equal: ===
- A veces no se trata de ser exactamente igual

```
if /sera/ === "coursera"
  puts "Triple Equals"
end
# => Triple Equals

if "coursera" === "coursera"
  puts "también funciona"
end
# => también funciona

if Integer === 21
  puts "21 es un Integer"
end
# => 21 es un Integer
```

Expresiones Case

- Se usa de dos formas:
  - **PRIMERA:** Similar a una serie de sentencias if / Estableciendo la condición siempre.
  - **SEGUNDA:** Especificando un target al lado de las sentencias **case** y cada cláusula **when** es comparado con el target.
- === es utilizado en el caso de igualdad
- En caso de matchear una sentencia when, esa es la única que se ejecuta, no hay forma de pasar al siguiente when (como en C/C++)

Expresión Case - Segunda forma

```
name = 'Fisher'
case name # 2nd FLAVOR
  when /fish/i then puts "Something is fishy here"
  when 'Smith' then puts "Your name is Smith"
end
```

EJERCICIO: Utilizar la PRIMERA forma de case.

edad = 30

Imprimir el texto “Eres Jóven” cuando la edad sea menor a 40.

Imprimir el texto “Eres un niño” cuando la edad sea menor a 30.

Ejercicio Resuelto

```
case # 1ST FLAVOR
  when edad < 40
    puts "Eres Jóven!"
  when edad <= 35
    puts "Eres un niño"
  else
    puts "No ingreso en ninguna opción del case!"
end
```

Solo se ejecuta el primer case

Flujo For

```ruby
for i in 0..2
  puts i
end

# => 0
# => 1
# => 2
```

Se utiliza en raras ocasiones, es más común ver each/times.

Conclusiones

- Existen muchas/todas opciones de flujo de control
- La forma inline o modifier es una manera muy expresiva de programar.
- Los objetos no-nulos y no-falsos son evaluados siempre como true.