ECI07
Charla de Baufest

Ruby

¡una joya!

baufest. (del alemán). *bau: construcción / fest: sólida*

baufest

¿Quiénes somos?

baufest

www.baufest.com

Oradores:

Lic. Diego Figueira

Lic. Alejandra Alfonso

2

- Empresa especializada en Ingeniería de Software
- Capitales Argentinos
- Fundada en 1991: 16 años
- Oficinas en Argentina, España y próximamente México
- 300+ proyectos en 30 países América, Europa, Asia y África
- 120+ personas

baufest

Agenda

- **Ruby**, ¿y eso con qué se come?
- Features interesantes del lenguaje.
- Syntactic sugar: lo que Ud. Siempre quiso hacer y su lenguaje no le permitió.
- **Ruby on Rails**, un web framework tan simple como es posible, pero no más.
- Mini-demo.

baufest

3

Contexto

- Creado originariamente por Yukihiro Matsoumoto (aka **Matz**) en **1993** como una mejora de Perl. Desde hace tiempo que es un lenguaje muy popular en Japón.
- Combina sintaxis inspirada en Perl y Smalltalk.
- Es **interpretado** en una pasada (no se compila).
- Existen implementaciones de su intérprete en Java, .Net y Smalltalk.
- Existe en varias plataformas: Windows, Linux, MacOS
- Es **Open Source** (GNU/GPL).

baufest

4

Qué tiene de especial?

- Paradigma: orientado a objetos puro.
- Flexible: tipado dinámico, reflexivo.
- Sintaxis amigable y limpia: fácil de aprender y comprender.
- Muy potente y expresivo.
- Robusto: usa garbage collector.
- Con una librerías de soporte ricas y abiertas.
- Productivo: desarrollo eficiente.

baufest

5

Ruby.new

baufest

6

Objetos

Ruby, un objeto con clase

- **Todo** es un objeto

Los enteros son objetos

```
-5.abs
⇒  5
42.zero?
⇒  false
('a'..'e').to_a
⇒  ["a", "b", "c", "d", "e"]
```

Todos los objetos tienen disponible mucha funcionalidad built-in

```
Object.new.methods.sort
⇒  ["==", "===", "=~", "__id__", "__send__", "class", ...]
```

```
nil.nil?
⇒  true
nil.class
⇒  NilClass
```

Pero eso ya estaba en Smalltalk!

¡Y nil también es un objeto!

baufest

7

Strings

ybuR, ¡me da vuelta la cabeza!

- **Interpolación de expresiones**:

```
n = "Pipi"
"mi nombre es #{n}"
⇒  "mi nombre es Pipi"
```

- Dos formas de **delimitación** de literales: **"** y **'**

```
'hoy es #{Time.now}' == "hoy es #{Time.now}"
⇒  false
```

- Son fáciles de usar:

```
saludo = "hola"
saludo.empty?
⇒  false
saludo.capitalize!
⇒  "Hola"
saludo.reverse
⇒  "aloH"
```

baufest

8

Expresiones regulares

Ruby, el día que Smalltalk lloró

Literales delimitados con **/ /**

```
er = /ab+c/
er.class
⇒ Regexp
```

Operador de *matching* **=~**: evalúa a *nil* si no hay match, o al índice de la primera aparición de la ER.

```
/(pi)+/ =~ "Mi nombre es Pipi"
⇒  15
```

Se puede recuperar el contexto del *match*:

```
re = /(\d+):(\d+)/
re =~ "La hora es 5:35 pm"
$&              ⇒ "5:35"
[$1, $2]        ⇒ ["5", "35"]
$`              ⇒ "La hora es "
$'              ⇒ " pm"
```

Las variables que empiezan con **$** son globales

Esta misma info está almacenada en un objeto **MatchData**

Esto se parece a Perl..!

9

Bloques e iteradores

```
[1,2,3].each {|i| puts i+1}
2
3
4
```

Los iteradores quedan elegantes, simplemente pasamos un bloque al método **each**

Los bloques son construcciones del lenguaje, no son **objetos**

```
a = [1, 2, 3, 4, 5, 6, 7]
a.select {|n| n % 2 == 0 }
⇒  [2, 4, 6]
a.inject do |s,n|
  s + n
end
⇒  28
```

Heredan el contexto en donde fueron creados.

El keyword **lambda** los convierte a objetos

```
def n_times(thing)
  lambda {|n| thing * n}
end
p = n_times(4)
p.call(3)
⇒  12
```

baufest

10

Métodos

Ruby, ¿Smalltalk++?

parámetros con valor default

```
class Saludo
  def saludar_a (nombre='pipi')
    puts "Hola #{nombre}!"
    "#{nombre} fue saludado"
  end
end
```

el último valor es el devuelto por la función

```
Saludo.new.saludar_a 'Mundo'
Hola Mundo!
⇒ "Mundo fue saludado"
```

¿Invocación de un método o **envío de mensajes**?

```
class Roman
  def romanToInt(str)
    # ...
  end
  def method_missing(methId)
    str = methId.id2name
    romanToInt(str)
  end
end

r = Roman.new
r.iv          ⇒ 4
r.xxiii       ⇒ 23
r.mm          ⇒ 2000

r.send("ix") ⇒ 9
```

11

Más sobre métodos

puede recibir una cantidad variable de argumentos

```
def varargs(*args)
  "Llegó #{args.join(', ')}"
end
```

```
varargs('un', 'dó', 'tré')
⇒  "Llegó un, dó, tré"
```

```
def dame_un_par
  return 55, 'pipi'
end
```

```
a, b = dame_un_par
a       ⇒ 55
b       ⇒ "pipi"
```

Pueden devolver múltiples objetos

¿Asignación paralela?

¡Sí! Como en:

```
a, b = b, a
```

12

Mixins

Ruby, el mejor amigo del programador

```
module Comparable
  def ==(arg0)
    ...
  end
  def >=(arg0)
    ...
  end
  def <(arg0)
    ...
  end
  def <=(arg0)
    ...
  end
  def >(arg0)
    ...
  end
  def between?(arg0, arg1)
    ...
  end
end
```

```
class Persona
  include Comparable
  ...
  def initialize(nombre, apellido)
    @nombre = nombre
    @apellido = apellido
  end
  def <=>(other)
    self.apellido <=> other.apellido
  end
end
```

La clase receptora debe implementar **<=>**

```
ale = Persona.new('Ale', 'Alfonso')
diego = Persona.new('Diego', 'Figueira')

ale > diego     ⇒ false
ale <= diego    ⇒ true
```

…y **hereda** todos los otros métodos de comparación

baufest

13

Azúcar sintáctica!

(Syntactic Sugar)

baufest

14

Arrays

Rubí or not Rubí

- Pueden pensarse como diccionarios con números enteros como clave.
- Son de tamaño dinámico.
- Pueden contener cualquier objeto en cada posición (no-tipados).

```
a = [1, 'pipi', 0.1]
a[1]          ⇒ "pipi"
a.length      ⇒ 3
a[5]          ⇒ nil
a[5] = "coco"
a.length      ⇒ 6
a             ⇒ [1,"pipi",0.1,nil,nil,"coco"]
```

El objeto **nil** representa la ausencia de elemento.

Longiutud dinámica.

- Muchas formas de acceso a múltiples elementos a la vez.

```
a[-1]
⇒  "coco"
a[0..2]
⇒  [1, "pipi", 0.1]
a[1, 3]
⇒  ["pipi", 0.1, nil]
```

[-n]: n-ésimo elemento contado desde atrás

[principio..fin]

[inicio, cantidad]

Asignación simultánea a varios elementos

```
a[3..5] = []
a      ⇒ [1, "pipi", 0.1]
a[0..1] = 'a', 4
a      ⇒ ["a", 4, 0.1]
```

baufest

15

Atributos

Ruby, joya nunca taxi

```
class Persona
  def apellido=(apellido)
    @apellido = apellido
  end
  def apellido
    @apellido
  end
  def nombre=(nombre)
    @nombre = nombre
  end
  def nombre
    @nombre
  end
end
```

=

```
class Persona
  attr_accessor :apellido, :nombre
end
```

```
juan = Pesona.new
juan.nombre = 'Juan'
juan.apellido = 'Perez'
"#{juan.apellido.upcase}, #{juan.nombre}"
⇒  "PEREZ, Juan"
```

baufest

16

Expresiones booleanas

Bienvenidos al planeta Ruby

```
def selected_products
  if @selected_products.nil? then
     @selected_products = []
  end
  return @selected_products
end
```

=

```
def selected_products
  @selected_products = @selected_products || []
  @selected_products
end
```

=

```
def selected_products
  @selected_products ||= []
end
```

Tabla del ||:

false	||	b	⇒	b
nil	||	b	⇒	b
a	||	b	⇒	a

17

baufest

Rails

RoR: ¡subite al tren de la alegría!

- Framework **extremadamente productivo** para construir aplicaciones Web
- Desarrollado por David Heinemeier Hansson
- Framework *full stack*: brinda soporte en todas las capas de la aplicación
- Basado en el principio **DRY**: *Don't Repeat Yourself!*
- *Convention over configuration*: menos configuración en favor de convenciones de nombres y *reflection*
- Basado en el pattern **MVC**:
  - *Model*: mapeo ORM, validaciones, asociaciones, transacciones, etc.
  - *View*: templates con scripts embebidos, layouts, soporte para Ajax.
  - *Controller*: ruteo de requests, filtros, cacheo y manejo de sesión.

19

baufest

Demo

20

baufest

Material

Ruby

http://www.ruby-lang.org/
Página oficial

http://pine.fm/LearnToProgram/
Tutorial minucioso

http://poignantguide.net/ruby/
Introducción al lenguaje, tan interesante como delirante

Ruby on Rails

http://www.rubyonrails.org/screencasts
¡Cuidado! Muy motivante.

http://aptana.com/
IDE para desarrollo web basado en Eclipse. Plugin para RoR 'RadRails'

http://wiki.rubyonrails.org/rails
Documentación y muchos recursos

http://www.softwaredeveloper.com/features/74-ruby-on-rails-resources-tutorials-050207/
Punteros a buenos tutoriales y recursos

Programming Ruby
The Pragmatic Programmers' Guide, Second Edition

Agile Web Development with Rails
(Pragmatic Programmers)

baufest

21

Muchas Gracias

Lic. Diego Figueira (dfigueira@baufest.com)
Lic. Alejandra Alfonso (aalfonso@baufest.com)

info@baufest.com
www.baufest.com

Argentina
Tel.: +54 (11) 4807-8080
Fax: +54 (11) 4801-6146
Av. Las Heras 3257
(C1425ASJ) Buenos Aires

España
Tel.: +34 91 745 2763
Fax: +34 91 561 5626
c/ Francisco Giralte, 2
(28002) Madrid

baufest

23