

“Collision Course with Ruby”



Gabriel E. Arellano / Marcos A. Caro

**Grupo de Estudios de Software Libre / Grupo de Estudios de UML
U.T.N. - F.R. Concepción del Uruguay**

¿De qué hablaremos?

- Introducción a Ruby
- Características del lenguaje
- Sintaxis elemental
- Elementos del lenguaje
- Herramientas
- Particularidades de Ruby
- Ejemplos, ejemplos, ejemplos!

El lenguaje Ruby

Un lenguaje de scripting que:

- Es orientado a objetos (puro).
- Tiene gran claridad semántica.
- Es fácilmente extensible y Open Source.
- Es dinámico e interpretado.
- Soporta reflexión y metaprogramación.
- Fue creado en Japón en 1995, y en pocos años desplazó casi totalmente a Python.

Sintaxis elemental

```
y = 3 + b
```

```
class Numeric
  def mas(x)
    self.+(x)
  end
end
```

```
y = 5.mas(6)
```

```
class Numeric
  def mas x
    self + x
  end
end
```

```
y = 5.mas 6
```

Todo es un objeto!

En Ruby todo es un objeto:

- Los objetos son objetos...
- Las clases son objetos...
- No hay operadores ni procedimientos...
- Los métodos son objetos (callable objects)
- Los bloques de código son objetos...
- Todos los objetos comparten ciertos atributos y métodos...

Ejemplo

Variables, constantes...

Existen distintos tipos de variables:

- Variables comunes: hora_dia, cantidad
- Variables de instancia: @hora_dia
- Variables de clase: @@hora_dia
- Variables globales: \$hora_dia
- Constantes: HoraDia, CANT
- Símbolos: :nombre, :Edad

Tipos de datos?

En Ruby no existen los tipos de datos, pero sí tenemos algunas clases que nos pueden resultar de utilidad:

- Numéricos
- Cadenas
- Literales
- Fechas y Horas

Estas clases no siempre se instancian explícitamente sino a través de asignaciones.

Ejemplo

Tipos estructurados

Son esencialmente dos:

- Arreglos

Son tipos estructurados de tamaño dinámico que cuentan con un índice numérico como mecanismo de acceso.

```
arr = [2, "hola", 3.2]
```

- Hashes

Similares a los arreglos pero emplean símbolos como mecanismo de acceso.

```
capitales = { :er => "Paraná", :cb => "Córdoba" }
```

Ejemplo

Métodos

Son el mecanismo “legítimo” y “cortés” para modificar el estado de un objeto...

- Son nombrados de la misma manera que las variables (pueden aparecer como atributos)
- Pueden recibir algún o algunos objetos como parámetros.
- Devuelven (o retornan) algún objeto.

Ejemplo

Clases

```
Class Cola
  def initialize
    @cola = Array.new
  end
  def inserta elemento
    @cola.push elemento
  end
  def extrae
    @cola.shift
  end
end
```

Ejemplo

Clases y Herencia

La herencia en Ruby tiene algunas particularidades:

- Sólo existe herencia simple.
- Se pueden incorporar métodos y constantes a través de módulos (mixin)
- Sólo se heredan métodos de clase.

Clases y Herencia

```
class ColaLimitada < Cola
  def initialize limite
    super()
    @max = limite
  end
  def inserta elemento
    if @cola.size < @max
      super elemento
    end
  end
end
```

Ejemplo

Accessors

En el paradigma de Orientación a Objetos, los atributos de un objeto sólo pueden ser modificados a través métodos del propio objeto.

Es bastante tedioso tener que agregar un par setter / getter por cada atributo...

Un ejemplo...

```
class ColaLimitada < Cola  
  
  attr_reader :cola, :max  
  attr_writer :max  
  
  ...  
  
end
```

Bloques e Iteradores

En Ruby todo es un objeto, por ello, inclusive los bloques de código tiene métodos...

Estructuras de Control y Condicionales

Además de los iteradores Ruby cuenta con las estructuras de control y condicionales típicas de otros lenguajes:

- Bucles do, for, while, until...
- Condicionales if, unless, when y case

Ruby es dinámico...

Ruby es un lenguaje dinámico:

- Puedo agregar atributos y métodos de manera dinámica...
- Métodos de clase y métodos de instancia.
- La clase y los métodos Singleton.
- Los métodos eval (ejecutar código almacenado en una cadena de texto)
- Callable methods y funciones anónimas.

Ejemplo

Ruby es Reflexivo...

Ya que la clase a la que pertenece cada objeto se define en tiempo de ejecución resultarían muy útiles métodos que permitan obtener información acerca de un objeto o de la clase a la que pertenece...

Ejemplo

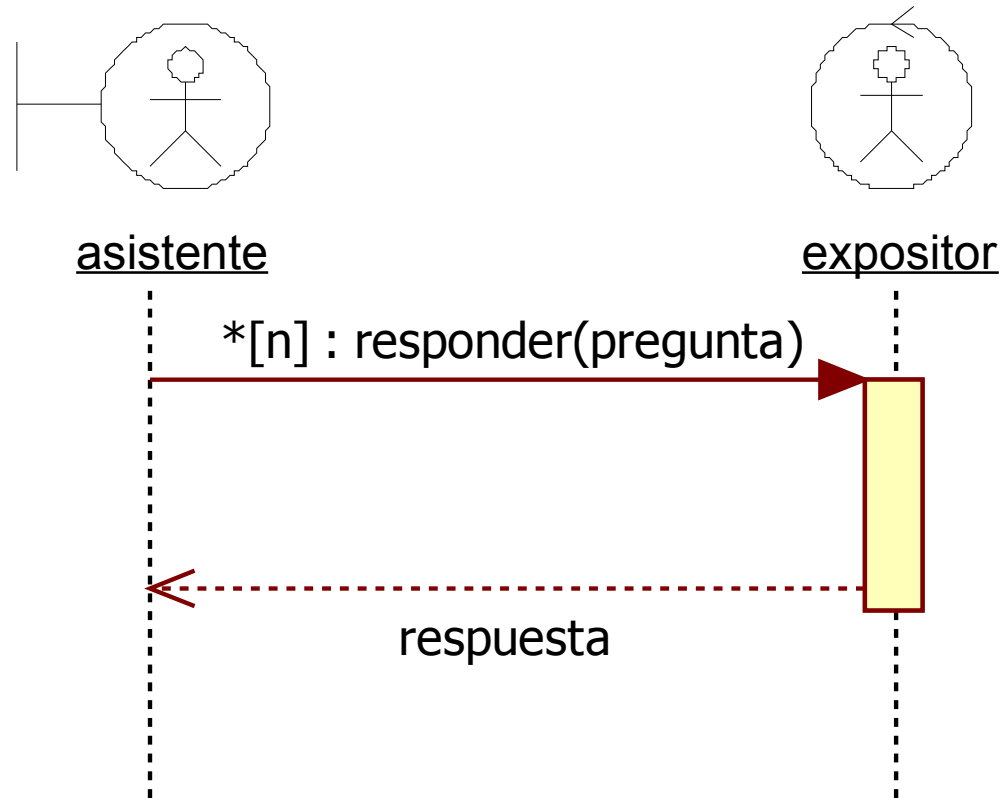
The “Ruby Way”

Ruby no me obliga a hacer las cosas de una manera determinada... puedo programar como lo hacía en mi lenguaje preferido o lo puedo hacer “a la Ruby” (Ruby Way)...

Porqué usar Ruby?

- Es un lenguaje OO “de verdad”.
- Permite alto nivel de abstracción.
- Me deja programar a mi manera...
- Es dinámico, reflexivo y facilita la metaprogramación y las técnicas de desarrollo ágil...

Preguntas?



Lecturas Recomendadas

“Programming Ruby” (2da. Edición)

Dave Thomas - Ed. The Pragmatic Programmers (2005)

<http://www.rubycentral.com/book/>

<http://pickaxe.ruby.org.es/>

“Ruby for Rails: Ruby techniques for Rails developers”

David A. Black. - Ed. Manning (2006)

Por dónde empezar?

Sitio fundamental!

<http://www.ruby-lang.org/es/>

Lista de Correo Ruby Argentina

<http://rubyargentina.soveran.com/signup>

Divulgación

<http://www.rubycorner.com/>

<http://www.rubycorner.com/blogs/lang/es> (español)

<http://rubyargentina.soveran.com/>

Créditos

Partes de la estructura y contenidos de esta charla fueron “inspirados” en el excelente curso de Brian Schröder disponible en:

<http://ruby.brian-schroeder.de/course/>

Gracias!

gabrielarellano@gmail.com
marcos.antonio.caro@gmail.com

<http://www.gabriel-arellano.com.ar/charlas/>

(2006) Gabriel E. Arellano

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. The GNU Free Documentation License as applicable to this document can be found at: <http://www.gnu.org/copyleft/fdl.html>