



**ESCUELA SUPERIOR
POLITÉCNICA DEL LITORAL**

Tema: Investigación de Lenguajes

Materia: Lenguajes de Programación

Nombre del grupo en Sidweb: Lenguaje Ruby

Paralelo: 1

Fecha de entrega:
Miércoles, 23 de Octubre del 2013

Lenguaje Ruby

Autores:

- Cristina Barreno
- Sixto Castro
- Jordy Vasquez

21 de octubre de 2013

Tabla de Contenidos

1. Introducción
2. Características
3. Historia
4. Tutorial de instalación
5. Hola Mundo y otros programas introductorios
6. Referencias bibliográficas

Lenguaje Ruby



Figura 1: Logo de Ruby

1. Introducción

Ruby Lenguaje creado por Yukihiro Matsumoto, combina lo mejor de la orientación a objetos (smalltalk) y la facilidad del scripting (perl) generando un lenguaje dinámico, muy expresivo, potente, fácil de aprender y que permite crear aplicaciones empresariales robustas, estables y seguras ademas de ser multiplataforma , es decir puede correr en gran cantidad de arquitecturas , hasta en telefonos celulares, y su implementacion es bajo licencia de software libre .

2. Características

- Ruby es un lenguaje multiplataforma (altamente portable).
- Es fácil de escribir.
- Es un lenguaje interpretado.
- Es un lenguaje orientado a objetos. Al igual que Java, cada tipo de dato es un objeto.
- Es una mezcla de varios lenguajes tales como Perl, Python, Smalltalk y otros, que definen a Ruby como un lenguaje que integra la programación funcional e imperativa.
- Posee expresiones similares(nivel de lenguaje) a las de Perl
- En Ruby, los métodos se pueden o no ser parte de una clase. Puede ser declarado en cualquier parte del archivo.
- Es capaz de manejar excepciones.
- Se puede manejar hilos (multihilos) sin depender del sistema operativo.

3. Historia

Yukihiro Matsumoto crea Ruby en Japon en el año 1993, mientras trabajaba en lenguajes como Perl y PHP, pero se ve tentado en vez de crear un Perl mejorado, su propio lenguaje combinando asi: **Perl, Smaltalk, Eiffel y Lisp**. Su nombre proviene como referencia al lenguaje Perl(perla) el cual originalmente deseaba mejorar.

Se lanza al publico en el año 1995, trayendo asi desarrolladores de todo el mundo, que ven a a Ruby como un lenguaje de exploracion. Tiene un crecimiento lento pero seguro hasta que David Heinemeier Hansson crea el framework Rails , el cual impulsa su crecimiento llevandolo a ser dominado como el **"Lenguaje de Progra maccion del 2006"** e instalandolo entre los 10 mas populares actualmente.

4. Tutorial de Instalación

Como bien sabemos ruby es un lenguaje multiplataforma por lo que aprenderemos a instalrlo en los diferentes sistemas operativos, asi mismo en todos los casos instalaremos la versión de Ruby **1.9.3** la cual es la version más estable actualmente, ademas te ofrece una extensa lista de paquetes o gemas compatibles y actualizadas.

■ Ruby en Windows

Para instalar el interprete de Ruby en Windows los podemos hacer desde su página oficil visitando <http://rubyinstaller.org/downloads/> donde descargaremos el *RubyInstaller* versión *Ruby1,9,3 – p448* para una arquitectura de 32 bits, si la arquitectura es de 64 bists puedes descargar la versión *Ruby2,0,0 – p247(x64)*.

Después que descarguemos el instalador lo ejecutamos como administrador y empezamos la instalación.Aceptamos el contrato de licencia.Vea *Figura2*

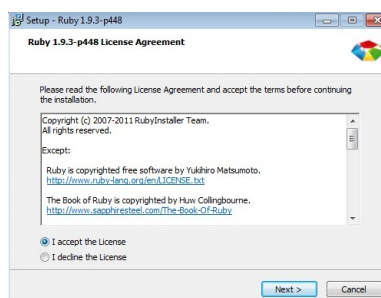


Figura 2: Paso de instalación 1

Agregamos lo necesario para la instalación como se observa en la *Figura3*

Terminamos la instalación. Vea *Figura4*

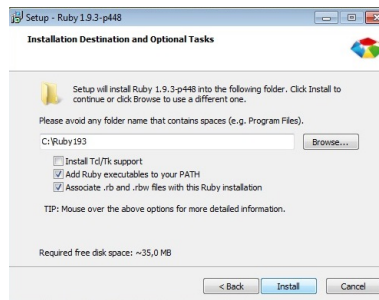


Figura 3: Paso de instalación 2

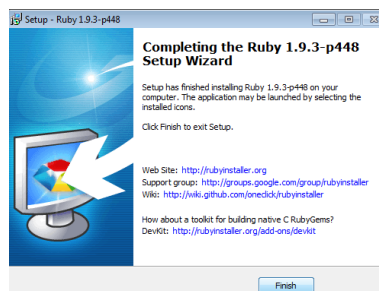


Figura 4: Paso de instalación 3

Abrimos *InteractiveRuby* y comenzamos a trabajar con Ruby desde consola. Vea *Figura5*

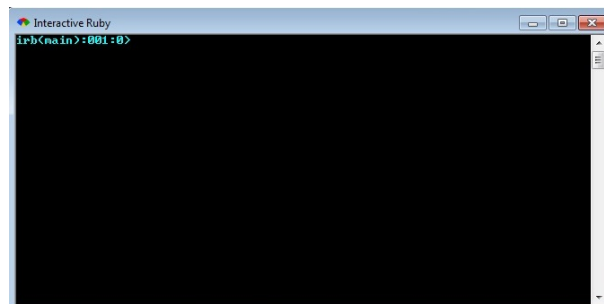


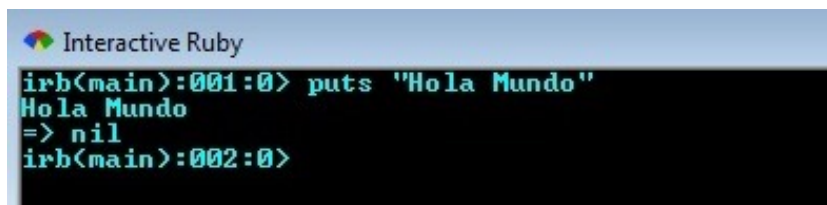
Figura 5: Paso de instalación 4

5. Hola Mundo y otros Programas Introductorios

A continuación revisaremos algunos programas básicos en ruby para esto solo necesitamos tener instalado en nuestro computador el interprete de ruby, abramos el terminal *InteractiveRuby* y empecemos a programar. Aunque solo trabajaremos desde consola recordemos que esta es una herramienta poderosa con la cual daremos nuestros primeros pasos en el mundo de ruby.

Ejemplo 1: Hola Mundo

Para hacer que nuestro programa escriba "*HolaMundo*" usaremos el comando *puts* "*HolaMundo*", siendo "*puts*" el comando básico para escribir algo en ruby.

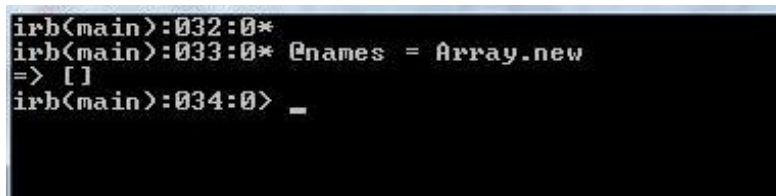


```
Interactive Ruby
irb(main):001:0> puts "Hola Mundo"
Hola Mundo
=> nil
irb(main):002:0>
```

Figura 6: Entrada del comando usado y su salida por consola

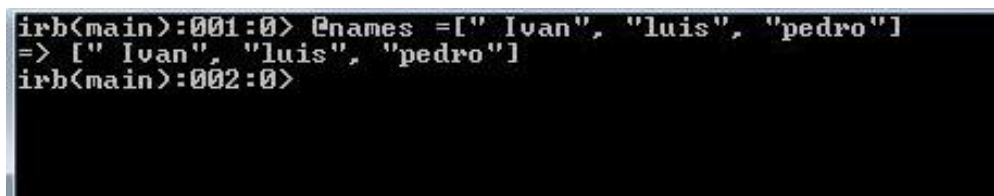
Ejemplo 2: Trabajando con arreglos y Strings

- Crear un array



```
irb(main):032:0*
irb(main):033:0* @names = Array.new
=> []
irb(main):034:0> _
```

- Método []



```
irb(main):001:0> @names = ["Ivan", "luis", "pedro"]
=> ["Ivan", "luis", "pedro"]
irb(main):002:0>
```

Crea el Array y lo inicializa con strings

- Método: +

```

irb(main):001:0> clientes2 = [ "Pierre", "Joe", "Rachel"]
=> ["Pierre", "Joe", "Rachel"]
irb(main):002:0> clientes = ["Juan", "Luis", "Roberto"]
=> ["Juan", "Luis", "Roberto"]
irb(main):003:0> clientes + clientes2
=> ["Juan", "Luis", "Roberto", "Pierre", "Joe", "Rachel"]
irb(main):004:0>

```

Método para unir un array al final de otro.

- Ordenar un arreglo

```

irb(main):001:0> notas=[2,4,3,6,8,2,3,5,1]
=> [2, 4, 3, 6, 8, 2, 3, 5, 1]
irb(main):002:0> notas.sort
=> [1, 2, 2, 3, 3, 4, 5, 6, 8]
irb(main):003:0>

```

- Obtener el tamaño de un arreglo

```

irb(main):001:0> notas=["12","16","9","17","19"]
=> ["12", "16", "9", "17", "19"]
irb(main):002:0> notas.size<
=> 5
irb(main):003:0> _

```

- Recorrer un arreglo: each

```

irb(main):001:0> equipos = ["Barcelona","Milan","PSG","Monaco"]
=> ["Barcelona", "Milan", "PSG", "Monaco"]
irb(main):002:0> equipos.each do |equipo|
irb(main):003:1* puts equipo
irb(main):004:1> end
Barcelona
Milan
PSG
Monaco
=> ["Barcelona", "Milan", "PSG", "Monaco"]
irb(main):005:0>

```

El método 'each' lo que va hacer es recorrer cada elemento del arreglo equipos. En cada iteración, cada elemento del array se guarda en la variable equipo y se va imprimir cada elemento(cadena) del arreglo equipos ya que se encuentra la función puts que se encarga de imprimir cada cadena.

- Recorrer el arreglo: for in

```
irb(main):017:0* for i in equipos
irb(main):018:1> puts i
irb(main):019:1> end
Barcelona
Milan
PSG
Monaco
=> ["Barcelona", "Milan", "PSG", "Monaco"]
irb(main):020:0> _
```

Este caso es parecido al ejercicio anterior, sino que ahora se utiliza un ciclo 'for' para recorrer el arreglo. En cada iteración, la variable i toma el valor actual del array equipos, y luego va imprimir dicho elemento ya que se hace un put en cada iteración.

- Igualdad entre arreglos

```
irb(main):020:0*
irb(main):021:0* alumnos="Jorge","Luis","Jordy"
=> ["Jorge", "Luis", "Jordy"]
irb(main):022:0> alumnos2="Marco","Orlando","Gabriel"
=> ["Marco", "Orlando", "Gabriel"]
irb(main):023:0> alumnos==alumnos2
=> false
irb(main):024:0> alumnos==["Jorge","Luis","Jordy"]
=> true
irb(main):025:0>
```

Devuelve false si no son iguales y true si lo son.

Ejemplo 3: objeto y metodos

- Ejemplo de crear un objeto y un metodo

```

irb(main):042:0*
irb(main):043:0* class Ver_Rang
irb(main):044:1> def initialize(caso=0)
irb(main):045:2> @caso=caso
irb(main):046:2> end
irb(main):047:1> def ver
irb(main):048:2> case @caso
irb(main):049:3> when 1, 2..5
irb(main):050:3> print "el numero esta dentro del rango 1 a 5"
irb(main):051:3> when 6..10
irb(main):052:3> print "el numero esta dentro del rango 6 a 10"
irb(main):053:3> else
irb(main):054:3> print "el numero no esta dentro del rango"
irb(main):055:3> end
irb(main):056:2> end
irb(main):057:1> end
=> nil

```

Las palabras claves son class para darle el nombre a nuestro objeto, def initialize para inicializarlo , y def ...(y el nombre del metodo) para crear un metodo para el objeto

Utilizando el objeto y su metodo

```

irb(main):073:0* Numero=Ver_Rang.new(5)
=> #<Ver_Rang:0x2c1b118 @caso=5>
irb(main):074:0> Numero.ver
el numero esta dentro del rango 1 a 5=> nil
irb(main):075:0> Numero2=Ver_Rang.new(9)
=> #<Ver_Rang:0x2c1bb80 @caso=9>
irb(main):076:0> Numero2.ver
el numero esta dentro del rango 6 a 10=> nil
irb(main):077:0> Numero3=Ver_Rang.new(33)
=> #<Ver_Rang:0x2bf4a18 @caso=33>
irb(main):078:0> Numero3.ver
el numero no esta dentro del rango=> nil
irb(main):079:0> _

```

Hacemos una instancia del objeto poniendo el [nombre del objeto].new, e inicializandolo ya sea con vacio o con un valor, y para utilizar el metodo de la misma forma [nombre del objeto].[nombre del metodo] como muestra Numero2.ver.

6. Referencias

- <https://www.ruby-lang.org/es/about/>
- http://es.wikibooks.org/wiki/Programaci3n_en_Ruby/Historia
- <http://www.maestrosdelweb.com/editorial/la-clase-array-en-ruby/>

- <https://www.ruby-lang.org/es/documentation/quickstart/2/>
- <http://eudev2.uta.cl/rid=1GR0DSG4D-1Y1NH87-4RQ/ruby.pdf>
- http://www.ecured.cu/index.php/Lenguaje_de_Programaci3n_Ruby
- <http://es.wikipedia.org/wiki/Ruby#Caracter.C3.ADsticas>
- http://es.wikibooks.org/wiki/Programaci3n_en_Ruby#Caracter.C3.ADsticas_Especiales_del_Lenguaje
- http://www.ecured.cu/index.php/Lenguaje_de_Programaci3n_Ruby#Caracter.C3.ADsticas_generales_del_lenguaje