Escuela de Informática y Sistemas

Jornadas de Administración e Informática
Año 2009

Conferencia: Ruby on Rails

Sergio A. Alonso
Analista en Sistemas
sergio@eim.esc.edu.ar
http://obelix.bunker.org.ar

Primer día: lunes 1 de junio

Introducción a Ruby - Demostraciones

Material

- Cañón proyector
- Computadora con Ubuntu provista por el disertante, instalada con el software necesario.
- Presentación en Openoffice Impress, disponible en http://www.campus.uner.edu.ar

Primer hora: Introducción a Ruby y a Rails

- Breve Historia
- Evolución Mundial
- Bindings
- Perpectivas a futuro

Segunda hora: Ruby: demostraciones mas o menos sorprendentes

Actividad: demostración en cañon proyector en el uso de:

- Irb: interactive Ruby o “el polígono de tiro”
- Todo como un objeto vs Variables primitivas
- For vs Iterar
- Fechas
- Arrays
- each vs for
- Matrices
- Hash
- MySQL: Demostración de unas pocas líneas en Linux

  Demostración en cañón proyector:

Todo como un objeto *vs* Variables primitivas

```
$ irb
Welcome to interactive ruby!

"saludos".length
=> 7
"saludos".methods
=> ... (muchos metodos)
"saludos".reverse.capitalize
=> "Sodulas"

3.times do
puts "holas "
end
holas
holas
holas

[3,1,7,0].sort.reverse
=> [7, 3, 1, 0]
```

For vs Iterar

```
for i in 1973..2008
puts i
end
1973
...
2008

1973.upto(2008) do | anio |
puts anio
break if i == 2007
end
```

```
2008
...
1973
```

Fechas

```
require 'date'
=> true
```

Llamamos y cargamos un *Módulo* de la librería base.
Módulo: colección de métodos, que no requieren de instanciación.

```
Time::now.year
=> 2007

sergio = Date.new(1973,2,26)

Time::now.year - sergio.year
=> 34

for i in sergio.year..2008
puts i
end

1973 .......... 2008

cristian = Date.new(1973,10,26)
cristian - sergio
=> Rational(242, 1)

cristian.month – sergio.month
=> 8
```

Arreglos

Nota: si observa el ejemplo, vera que pueden contener toda clase de elementos. Si lo que usted necesita es una matriz, existe también un objeto para ello (Matrix)

```
arreglo=["1",2,0.3],[4,5,6]
=> [["1", 2, 0.3], [4, 5, 6]]
matriz[1][1]
=> 5
lenguajes = ['Python', 'Java', 'Ruby']
lenguajes[1]
=> "Java"

lenguajes[0..2]
=> ["Java", "Python", "Ruby"]

lenguajes[0,2]
=> ["Java", "Python"]



for i in lenguajes
    puts i
end
=> ["Java", "Python", "Ruby"]

lenguajes << 'PHP'
=> ["Python", "Java", "Ruby", "PHP"]
```

bloques vs for

```
lenguajes = ['Python', 'Java', 'Ruby']

lenguajes.each do |lenguaje|
  puts 'Me gusta ' + lenguaje + '!'
end
=> Me gusta Python
=> Me gusta Java
=> Me gusta Ruby

for i in lenguajes
   puts "me gusta " + i
end

=> Me gusta Python
```

```
=> Me gusta Java
=> Me gusta Ruby
```

Diccionarios (en rigor: Hash)

Vectores con índices manuales como clave para encontrar un contenido

legajos = { "uno" => "Alonso" , "dos" => "Pacifico" , "tres" => "Ramirez" }

legajos[“dos”]

=> Pacifico

```
diccionario = { 'uno' => 1 , 'dos' => 2 }
=> {"uno"=>1, "dos"=>2}

diccionario['tres'] = 3
=> 3

diccionario
=> {"uno"=>1, "tres"=>3, "dos"=>2}

diccionario['uno']='otra cosa'
=> "otra cosa"

diccionario
=> {"uno"=>”otra cosa”, "tres"=>3, "dos"=>2}
```

Los símbolos como índices

Como indice de los hashes se pueden usar una expresiones llamadas Simbolos, las cuales vienen a ser un especie de etiqueta; una suerte de constante literal que no posee contenido.

```
persona = Hash.new
persona[:nombre] = 'Pedro'
persona[:apellido] = 'Picapiedra'

puts persona[:nombre]
```

que es equivalente a:

```
persona = {:nombre => 'Pedro', :apellido => 'Picapiedra'}
```

```
puts persona[:apellido]
```

Modelo try-catch-finally

```
F = File.open("archivo")
begin
   #...
   rescue
   #...
    else
   puts "No hubo errores"
ensure
  if not f.nil?
       f.close
end
```

Acceso a MySQL / MSSQL

-  En Linux requiere de agregar unas librerías:

  ```
  sudo aptitude install libmysql-ruby libdbd-mysql-ruby
  ```

-  En Windows, bajo **InstantRails** las librerías ya están incluidas.

Dentro del campus http://campus.uner.edu.ar, puede encontrar estos ejemplos en la sección de Archivos del curso: http://www.campus.uner.edu.ar/files/index.php?id=140

1) Fuentes/acceso_a_mysql.rb

```
require 'dbi'

dbh = DBI.connect('DBI:Mysql:inventario', 'root', '')

sth = dbh.prepare('select * from material')
sth.execute

while row=sth.fetch do
    p row
end

sth.finish
```

2) Fuentes/acceso_a_mssql.rb

```
require 'dbi'

DB = "Provider=SQLOLEDB.1;Password=coliflor;Persist Security
Info=True;User ID=sa;Initial Catalog=eith;Data
Source=192.168.1.118"

sql = " select * from Alumno"
DBI.connect("DBI:ODBC:#{DB}") do | dbh |
dbh.select_all( sql ) do |row|
     puts row[0] # the first field
     puts row[1] # the second field
end
end
```

Bindings gráficos

Se puede programar ventanas en Ruby? Por supuesto: Ruby puede utilizar diversos toolkits gráficos:

- Tk
- wxRuby
- GTK
- QT
- CocoaRuby (Objetive C, para Mac)
- Shoes

Uno de los bindings mas avanzados, estables, multiplataforma y mejor documentada, es **wxRuby**.

Atención: hay que tener las librerías wxruby instaladas (ver sección de instalación, día 2)

Veamos un ejemplo:

Primero: ¿donde están instaladas las gemas?, en mi caso, Ubuntu 8.04.x "Hardy", los consulto con:

```
gem environment
```

 En Linux anunciará que las **gemas** están en

```
/usr/lib/ruby/gems/1.8/gems
```

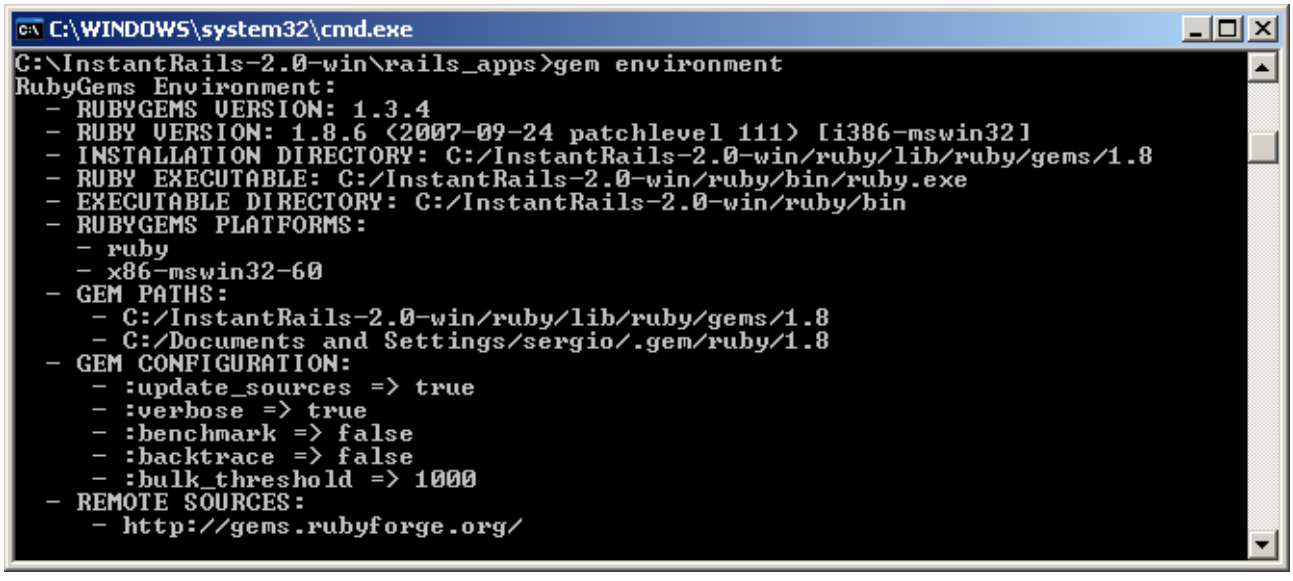
Es decir, que miramos un poco mas adentro, encontraremos una carpeta

```
wxruby-2.0.0-x86-linux
```

Con varios ejemplos adentro:

```
/samples/
```

 **Y en Windows ? Es lo mismo.**



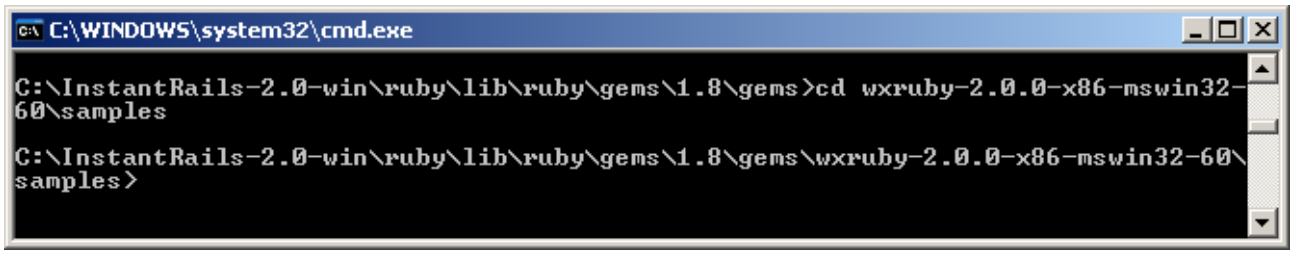
Es decir, podemos usar **CD** para llegar a la carpeta con las gemas:

```
cd \InstantRails-2.0-win\ruby\lib\ruby\gems\1.8\gems
```

Por curiosidad, para ver cuantos "**tesoros**" hay allí, podemos hacer un

```
dir
```

y entrar a la gema **wx** con sus ejemplos:



Ahora si:

Para probar los ejemplos, con código fuente y todo:

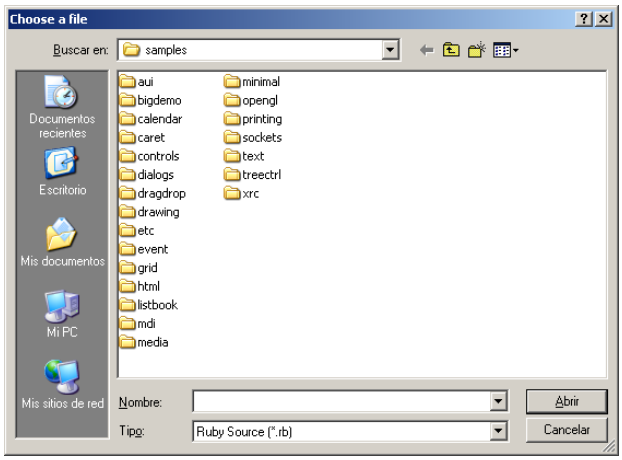
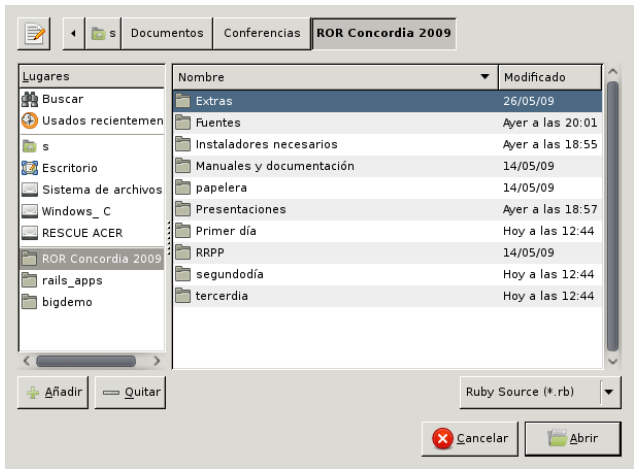
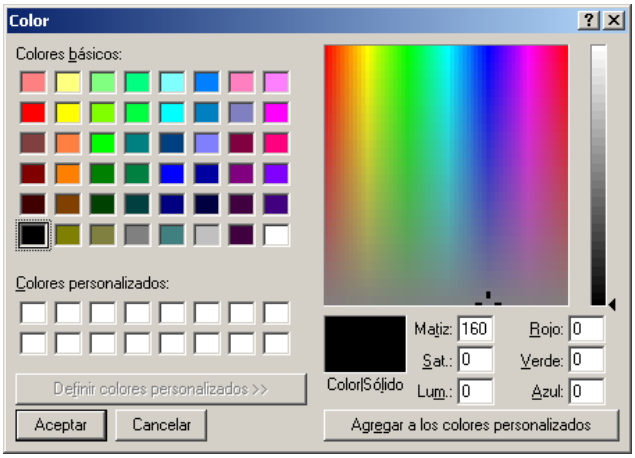
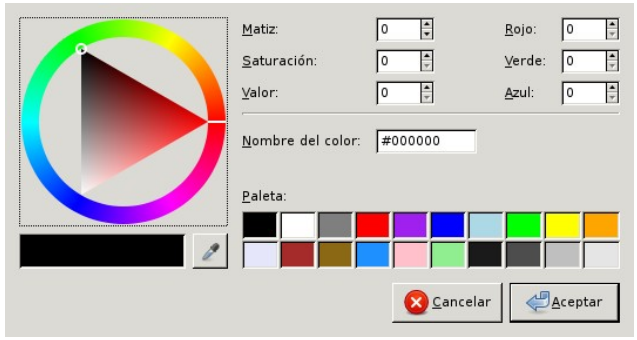
 `ruby bigdemo/bigdemo.rb`

Podemos copiar y adaptar el código fuente de los programas a nuestro gusto. El código que fabriquemos correrá igual en Linux, Unix, OS/X... hasta en Windows!

Curiosidad: observar como wxRuby implementa los elementos propios de ***cada*** sistema operativo, accediendo directamente a la API:

Ejemplo: observar dentro de la categoría Commons Dialogs, los elementos

- wxColourDialog
- wxFileDialog

Windows	Linux
	
	

Y en MAC OSX ?Ni idea, pero si desean saberlo, mi cumpleaños es el 26 de febrero.

Módulos

Definición: *Módulo*: colección de métodos, que no requieren de instanciación. Lo habíamos usado en el ejemplo

```
Time::now.year
```

¿Se puede crear módulos? ¡Por supuesto! Cumplen la función de las funciones en la programación estructurada.

```
#creamos un vector con cadenas
claves = ['python', 'java', 'ruby']

#solo por curiosidad, le agregamos un elemento
claves.push('eiffel')


def encriptar(v)
  #solo por curiosidad, ordenamos el vector enviado
  v.sort

  #creamos un nuevo vector
  vv = []

  #Representamos cada (each) elemento del vector con la variable i
  v.each do |i|

    #Ponemos el elemento en Mayúscula al principio
    i = i.capitalize

    #Lo damos vuelta
    i = i.reverse

    #Lo agregamos al vector nuevo
    vv.push i
  end

  return vv
end

#Probemos como quedó el módulo:
claves_encriptadas = encriptar(claves)

puts claves_encriptadas

Respuesta:
=> nohtyP
=> avaJ
```

```
=> ybuR
=> leffiE
```

Clases

Ejemplo de juego RPG con Ruby, con dos clases.

```
class Personaje
  #Atributos accesores (getters y setters pero con estilo)
  #attr_reader
  #attr_writer
  #attr_accesor

 attr_reader :nombre, :fuerza, :magia
  @nombre
  @clase
  @fuerza
  @magia

  #constructor
  def initialize(n,c,f,m)
    @nombre = n
    @clase = c
    @fuerza = f
    @magia = m
  end
end

class Juego

  def initialize(p1,p2)
    @personaje1 = p1
    @personaje2 = p2
  end



  def combate
    if (@personaje1.fuerza + @personaje1.magia) <
```

```
(@personaje2.fuerza + @personaje2.magia)
      puts "Gana " + @personaje2.nombre
    else
      puts "Gana " + @personaje1.nombre
    end
  end
end

#comienza juego
#escribo "main" para que se sienta cómodo :)

#instanciamos personajes
arturo = Personaje.new("Arturo","Caballero",10,5)
smaugh = Personaje.new("Smaugh","Dragon",50,40)

#instanciamos un juego nuevo
SeniorDeLosLadrillos = Juego.new(arturo,smaugh)

#Let's figth! (parafraseando a Mortal Kombat ;)
SeniorDeLosLadrillos.combate
```

Herencia

La herencia es muy simple. Utiliza el operador <

```
class JuegoV2 < Juego
end
```

Clases temporales

Si necesitamos momentáneamente un método de una clase, y nada mas, podemos instanciar una clase sin ocuparnos de pensar en un nombre, ni tener que destruirlo mas tarde.

```
Juego.new(arturo,smaugh).combat
```

Métodos Bang!

Hay ocasiones en que por cada método, encontramos uno similar terminado con un símbolo !

Estos símbolos modifican el objecto que los llama.

```
mensaje = 'Die die, piggy, die die!'
copia = mensaje.upcase
puts mensaje
>> Die die, piggy, die die!
puts copia
>> DIE DIE, PIGGY, DIE DIE!
```

En cambio con !

```
mensaje = 'Die die, piggy, die die!'
copia = mensaje.upcase!
puts mensaje
>> DIE DIE, PIGGY, DIE DIE!
puts copia
>> DIE DIE, PIGGY, DIE DIE!
```

¿Cuál es su utilidad? Evitar expresiones largas, como

```
mensaje = mensaje.upcase
```

Actividades Extras Recomendadas

- Probar el tutorial interactivo de Ruby, en http://tryruby.hobix.com
- Analizar el archivo Fuentes/ServerWebEnRuby.rb
- Analizar equivalencia a los atributos estáticos de Java, que en Ruby se usan como @@. Seguir el vinculo en http://mexicoonrails.com.mx/articles/metodos-de-clase-instancia-y-atributos-virtuales