Thin as a WEBrick

Adrian Kosmaczewski

2005-02-16

Estuve probando un nuevo lenguaje de programacion que no conocia: Ruby. Interesante pero medio criptico a veces; es un derivado de Perl (de ahi viene el nombre, de "perla" a "rubí") y de la familia del Algol (como todos los lenguajes "clasicos": C, C++, Java, PHP, etc).

Caracteristicas principales:

- Es un lenguaje de script interpretado, no compilado;
- Orientado totalmente a objetos; todo, incluso los numeros, son objetos;
- las instrucciones no se separan con punto y coma sino con nuevas lineas (como el BASIC);
- No tiene tipos (como JavaScript);
- No se necesita declarar las variables (identico a PHP pero diferente de JavaScript);
- Las variables no necesitan tener un "$" delante (como sucede en Perl y PHP);
- Tiene una sintaxis mas compleja de lo comun para los bucles;
- Tiene soporte nativo de expresiones regulares, obviamente (como JavaScript y Perl);
- Se pueden definir clases, con una sintaxis simple y "clasica" (keyword "class");
- Para definir funciones, se usa la palabra clave "def"; tambien para definir metodos dentro de las clases (como en Lisp, y con la misma posibilidad de definir funciones dentro de funciones, identico a Lisp y a JavaScript);
- Tiene una biblioteca de funciones de la hostia;
- La distribucion oficial pesa solamente (atencion!) 2 MB, nada mas, con todo incluido, es todo el codigo fuente!;
- Es open source y gratarola;
- Existe para cualquier sistema operativo, incluso Windows y Mac OS X, obviamente.

Se parece a esto:

```
# esta  linea con numeral es un comentario
# aca declaro y defino una variable string
str = "abc"
```

```ruby
puts str.length            # =>  3
# lo mismo para un array
arr = [10, 20, 30, 40, 50]
puts  arr.length           #  => 5
```

"puts" es la instruccion para "escribir" algo en la pantalla; "gets" es para leer. Nada demasiado complicado por ahora.

Tambien se pueden hacer cosas bastante complicadas, con dos lineas de codigo, usando una potente libreria que nada tiene que envidiarle a la de .NET, Java o Cocoa: por ejemplo abrir una conexion de red y leer informacion de un servidor remoto:

```ruby
require 'socket'
puts TCPSocket.open("server_name",  "daytime").read
```

Ruby fue creado por un japones que ya se volvio leyenda; esto fue hace unos 10 años mas o menos. Hay muchas librerias Externas disponibles en la web que se instalan facilisimo.

Para que sirve? Es un lenguaje de script, asi que se usa mucho para programacion web (en servidores mayormente); ultimamente se esta usando mucho como lenguaje de enseñanza, tambien para automatizar pruebas y tests.

En el Mac OS X viene instalada la version 1.6.8, pero la ultima version es la 1.8.2; haciendo unos tests de una revista me di cuenta que necesitaba la version 1.8.2, la baje (2 MB!), la compile y la instale y anduvo al toque. Tremendo.

En la revista Dr. Dobbs del mes pasado (enero 2005) aparece el siguiente script, que permite:

1. Crear un web server en el puerto 2000 en la maquina en la que funciona el script;
2. Cargar un "servlet" como los de J2EE, para que responda a los pedidos en la pagina "blog" devolviendo el contenido de todos los archivos HTML del sitio en cuestion;

He aqui el codigo, reproducido sin permiso ni nada, pero bueno, esperemos que nadie se chive, de todas maneras no gano un mango con esto:

```ruby
# este script funciona con ruby 1.8.2; aparece en la edicion de
# enero del 2005 de la revista "Dr. Dobb's Journal"
# esta linea indica que estamos importando la libreria "webrick"
require 'webrick'

# aca se define una clase que representa un articulo en un website;
# cada "articulo" existe fisicamente como un archivo HTML, por eso
# el "constructor" (el metodo "initialize") toma como parametro el
# nombre del archivo.
class Article
```

```
    # estas son las variables internas (campos) de la clase
    attr_reader :file_name, :title, :body

    # este es el metodo "constructor" de la clase,
    # que es llamado cada vez que se hace por ejemplo
    # "Article.new("index.html")"
    def initialize(file_name)
        body = File.read(file_name)
        title = file_name

        # aca se usan regexps para leer el archivo HTML...
        if body =~ %r{<title.*?>(.*?)</title}m
                || body =~ %r{<h1.*?>(.*?)</h1}m
            title = $1
        end
        body.sub!(%r{<body.*?>(.*)</body.*}m) { $1 }

        # le damos valores a los campos de la clase
        @file_name, @title, @body = file_name, title, body
    end
end

# definicion de un metodo de la clase "Article"
# grosso modo, este metodo hace lo siguiente:
# 1) saca la lista de archivos HTML que estan en el folder que se pasa en parametro
# 2) los ordena segun la fecha de creacion
# 3) para cada archivo encontrado, crea una nueva instancia de la clase "Article"
# la sintaxis para los bucles en ruby es alucinante!
def Article.list(dir)
    Dir.chdir(dir) do
        file_list = Dir.glob("**/*.html")
        sorted_list = file_list.sort_by {|name| File.stat(name).mtime }
        sorted_list.reverse[0, 10].map do |file_name|
            Article.new(file_name)
        end
    end
end

# el "<" indica una relacion de herencia con otra clase llamada
# WEBrick::HTTPServlet::AbstractServlet
class BlogServlet < WEBrick::HTTPServlet::AbstractServlet

    HEAD = "<head><title>Sample Blog</title></head>"

    def do_GET(request, response)
        articles = Article.list(@server.config[:DocumentRoot])
```

```
        content = articles.map{|a| a.body}.join("<" + "hr />")
        response.body = "<html>#{HEAD}<body>#{content}</body></html>"
    end
end

# primero creamos un webserver en el puerto 2000
server = WEBrick::HTTPServer.new(:Port => 2000, :DocumentRoot => "html")

# si el usuario tipea "CONTROL+C" el server tiene que cerrarse
trap("INT") { server.shutdown }

# aca cargamos el servlet en la memoria y arrancamos el servidor
server.mount("/blog", BlogServlet)
server.start
```

Todo esto se logra gracias a la libreria WEBrick que se baja de http://www.webrick.org/, y que, contrariamente al disco de Jethro Tull, no es “Thick as a Brick” sino mas bien “thin” ya que pesa solo 70 KB. Y el circulo se cierra, ya que estuve aprendiendo Ruby mientras escuchaba ese disco, justamente… y se explica el titulo pelotudo de este post que ya se hizo demasiado largo, como es costumbre. :)

Muy piola esto de Ruby, me gusto. Voy a seguir investigando…