Ruby y tecnologías

Ing. Sebastian M. Priolo

Bibliografía

- Documentación oficial
- Guía del usuario Ruby
- The Ruby Way (Hal Fulton)
- Ruby Developer's Guide
- The little book of Ruby (HuwCollingbourne)

Contenido

- Presentación de Ruby
- Ruby y las bases de datos: KirkyBase
- Ruby y los datos: YAML
- Ruby y GUI: Shoes

Ruby

- Lenguaje de scripts
- Multiplataforma
- Perl, Smalltalk, Eiffel, Ada y Lisp
- POO
- Sencillo
- Flexible
- Libre

Ruby

- Scripts
  - Automatización SO
  - Aplicaciones
  - Web
- Multiplataforma
- Orientado a objetos
- Lógica imperativa y funcional

Ruby

- Modificar todo
- Herencia Simple
- “imitar” herencia múltiple (módulos, mixin)
- No requiere declaración de tipos
- Hilos propios
- Recolector basura alto nivel

Ruby

- **Yukihiro Matsumoto**
- Perl ++
- el 21/12/95 Version 0.95
- el 25/12/96 Ruby 1.0 es lanzado al público
- Articulo de IBM
- 2004 Rails fue liberado. David Heinemeier Hansson
- Leng progr 2006. (TIOBE)

Ruby

- Son innecesarias las declaraciones de variables
- Las variables no tienen tipo
- La sintaxis es simple y consistente
- La gestión de la memoria es automática

Ruby

- No oculta las soluciones
- Cercano al dominio del problema
- No mapea al diseño constantemente
- Codificación mas rápida
- Comprensible
- Mantenible

Ruby

- Todo es un objeto
- Clases
- Herencia y métodos
- Métodos singleton
- Mixins por módulos
- Iteradores y cierres

Base de Datos

- DBI
- ODBC

- Oracle
- MySQL
- SQL Server
- SQLite
- ...

KirbyBase

- http://www.netpromi.com/

- SGBD
- Ruby
- Jamey Cribas
- SQLite

KirbyBase

- Multiplataforma
- Es pequeña y no necesita demasiados recursos
- Los datos se guardan en archivos planos y pueden ser modificados fácilmente.

KirbyBase

- Permite realizar consultas, actualizaciones, borrados.
- Utiliza distintos tipos de datos
- Permite guardar y restablecer objetos.
- Puede ser utilizada de forma remota.

Kirbybase

- una **base de datos** es un **directorio**
- **cada tabla** se **almacena en un archivo**
- **campo recno**

Conectar

```
require 'kirbybase'
base = KirbyBase.new
```

```
base = KirbyBase.new(:client,
'localhost', 44444)
```

Crear una tabla

```
usuarios = base.create_table()
```

Resumen

```
require 'kirbybase'
base = KirbyBase.new
usuarios =
base.create_table(:usuarios,:nombre,
:String,:apellido,:String)
clientes =
base.create_table(:clientes,:codigo,
:Integer,:pago,:Boolean)
```

- Files .tbl

Insertar datos

```
usuarios.insert('Jose','Moreno')
```

- Insertar registros desde un hash.

```
usuarios.insert(:nombre='Pablo',
:apellido='Filipini')
```

Insertar datos

- Bloques

```
usuarios.insert do |r|
    r.nombre = 'Carlos'
    r.apellido = 'Juarez'
end
```

Consultar

```
resultado = usuarios.select
```

```
resultado = usuarios.select(:nombre)
```

```
resultado =
usuarios.select(:nombre,:apellido){ |r|
r.nombre == 'Jose' }
```

Ordenar

```
resultado =
usuarios.select(:nombre)sort(:nombre)
```

```
resultado =
usuarios.select(:nombre)sort(-:nombre)
```

Reportes

```
puts usuarios.select.to_report
```

```
recno | nombre | apellido
------------------------
1 | Jose    | Moreno
2 | Pablo   | Filipini
3 | Carlos | Juarez
```

Reportes

```
puts usuarios.select.sort(:nombre).to_report
```

```
recno | nombre | apellido
-------------------------
  3 | Carlos | Juarez
  1 | Jose   | Moreno
  2 | Pablo  | Filipini
```

Actualizar

```
usuarios.update(:nombre=>'Carlos',
:apellido=>'Juarez') { |r| r.nombre ==
'Alberto' }
```

```
usuarios.update_all { |r| r.nombre =
'Prueba'}
```

Borrar

```
usuarios.delete { |r| r.nombre == 'Prueba' }
```

```
resultados = usuarios.pack
```

```
usuarios.clear
```

GUI

- El elemento visual
- El elemento información
- El elemento interacción

GUI

- Navegación simple y fácil.
- Posibilidad de identificar el espacio actual de trabajo.
- Reducción de la carga de memoria a corto plazo.
- Maximizar la cantidad de información.
- Prevenir errores
- Posibilidad de revertir acciones.

GUI

- Tk
- FXRuby
- wxRuby (wxWindows/wxWidgets)
- Ruby/Gnome2 (GTK)
- Ruby/Qt
- CocoaRuby
- Shoes

Shoes

- Minimalista
- Pocos elementos
- Multiplataforma
- Simple
- Clara

- http://code.whytheluckystiff.net/shoes/

Shoes

- Edit_line
- Edit_box
- Button
- Progress
- List_box

Ventana

```
Shoes.app :height => 250, :width => 200
do
end
```

Ventana

```
Shoes.app :height => 250, :width => 200,
:title => "Caso de Estudio 1" do
end
```

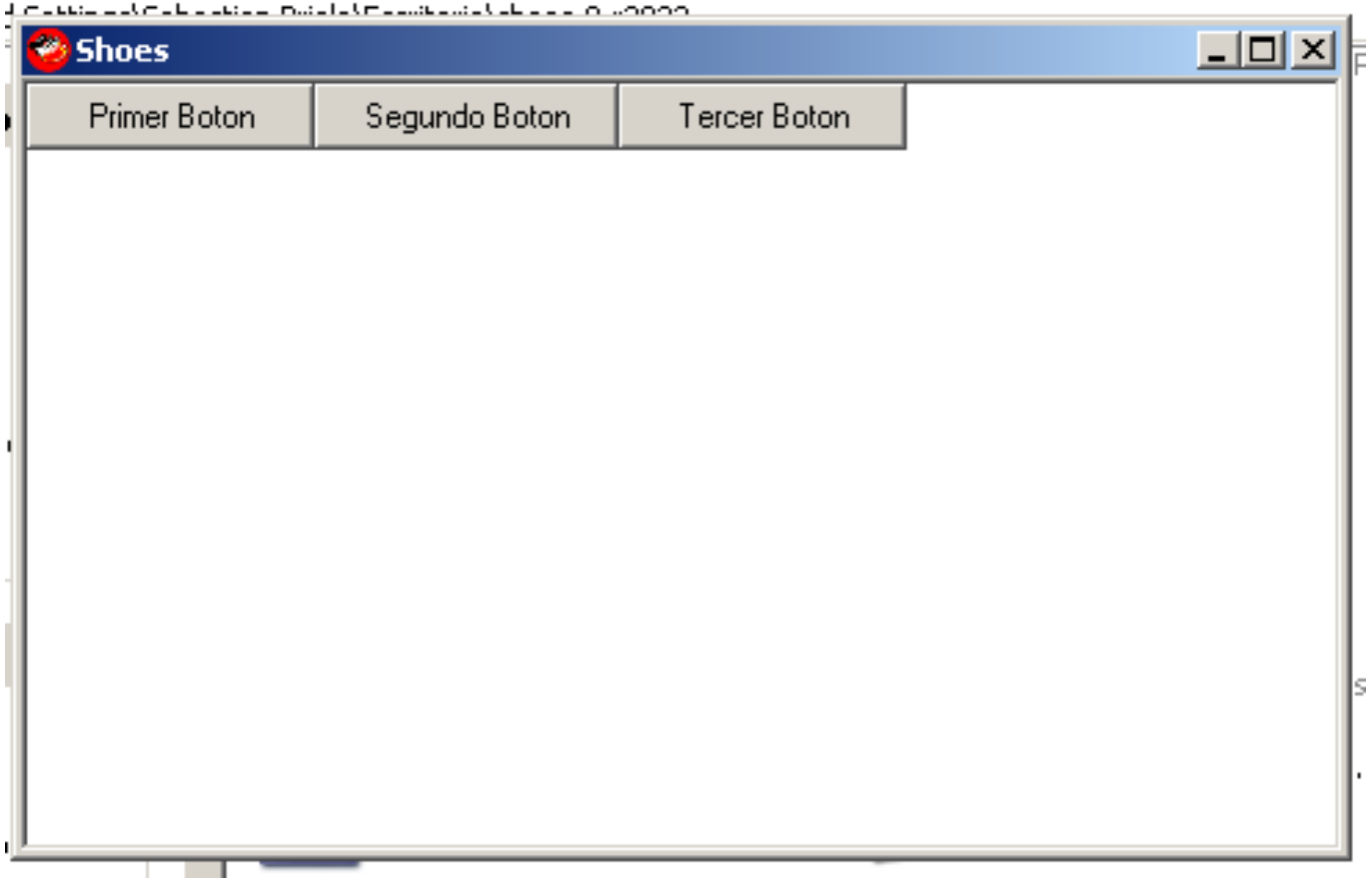
Botones

```
Shoes.app :height => 200, :width => 200
do
   button "Primer Boton" do
end
end
```

Botones

```
    Shoes.app :width => 640, :height =>
700, :title => "Caso de Estudio 2" do
    button "Primer Boton" do
    end
    button "Segundo Boton" do
    end
    button "Tercer Boton" do
    end
    end
```

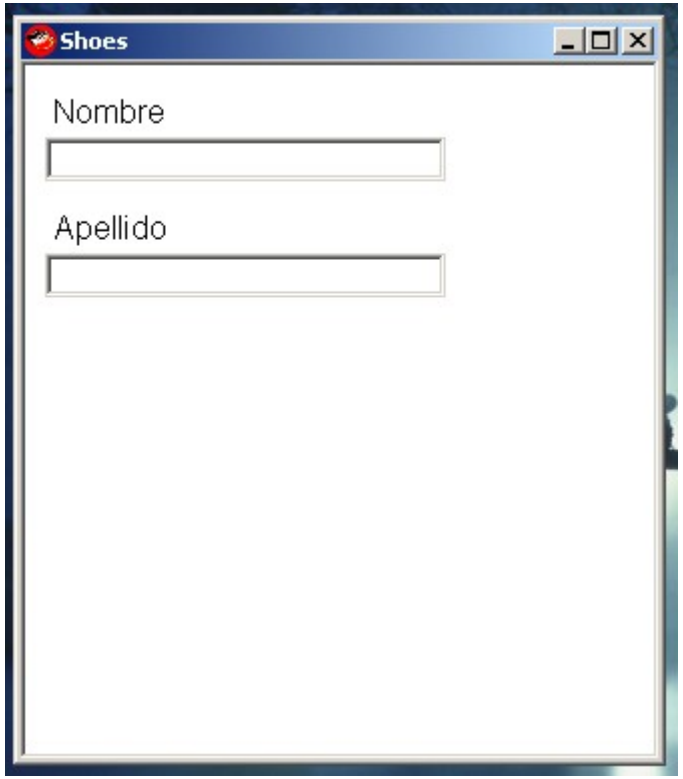
Botones

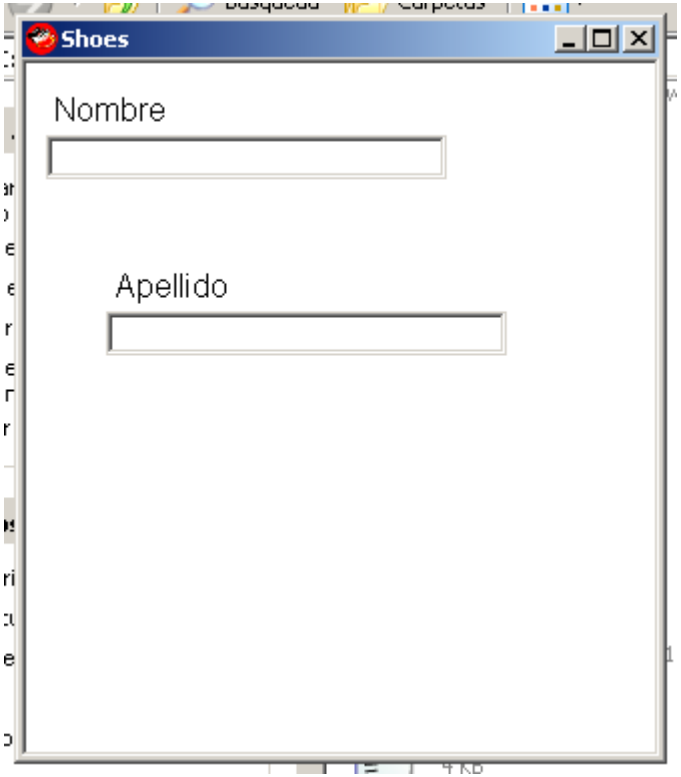


Campo de texto

```
Shoes.app :width => 320, :height => 350 do
      stack :margin => 10 do
         para "Nombre"
         @nombre = edit_line
      end
 stack :margin => 10 do
         para "Apellido"
         @apellido = edit_line
      end
end
```

Campos de texto





Stacks y Flows

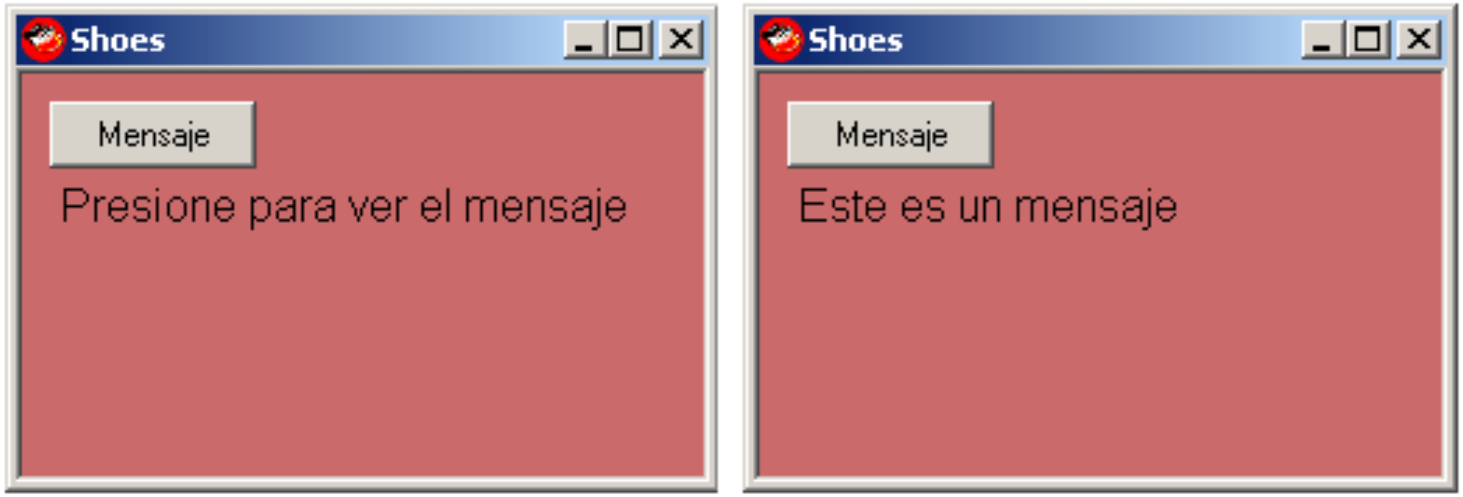
- stacks son contenedores que despliegan los elementos uno debajo de otro

- los flows despliegan los elementos de izquierda a derecha

Aplicación

```
Shoes.app :height => 150, :width => 250 do
  background rgb(240, 250, 208)
  stack :margin => 10 do
          button "Mensaje" do
            @etiqueta.replace "Este es un mensaje"
          end
           @etiqueta = para "Presione para ver el
mensaje"
  end
end
```

Aplicación


Shoes
Mensaje
Presione para ver el mensaje
Shoes
Mensaje
Este es un mensaje

YAML

- YAML Ain't Another Markup Language
- serialización de datos
- diseñado para ser legible
- inteligible por los seres humanos
- listas, hashes y valores simples

Características

- Los documentos son generalmente más pequeños que en XML.
- Fácil lectura y edición
- Presenta buena interacción con lenguajes dinámicos.
- Es altamente expresivo.
- Es extensible.
- Es de fácil implementación.

Documentos

Documento 1

Documento 2

Documento 3

Documento n

Secuencia simple

Elementos ordenados

- Elemento 1
- Elemento 2
- Elemento 3

Secuencia anidada

-
  - Elemento 1
  - Elemento 2
  - Elemento 3

-
  -
    - Elemento 1
    - Elemento 2

Mapeo simple y anidado

1: Elemento 1
2: Elemento 2

1: Elemento 1
2:
 2.1: Elemento 2.1
 2.2: Elemento 2.2
 2.3: Elemento 2.3

YAML y Ruby

```
require 'yaml'
class Mascota
attr_accessor :nombre, :raza, :edad
end
zeus = Mascota.new
zeus.nombre = "Zeus"
zeus.raza = "Ovejero Aleman"
zeus.edad = "3"
hera = Mascota.new
hera.nombre = "Hera"
```

YAML y Ruby

```
hera.raza = "Fox Terrier"
hera.edad = "4"
ares = Mascota.new
ares.nombre = "Ares"
ares.raza = "Labrador"
ares.edad = "5"
test_data = [ zeus,hera,ares ]
puts YAML::dump(test_data) #Con
esta instrucción
#hacemos el volcado a YAML
```

YAML y Ruby

```
---
- !ruby/object:Mascota
  edad: "3"
  nombre: Zeus
  raza: Ovejero Aleman
- !ruby/object:Mascota
  edad: "4"
  nombre: Hera
  raza: Fox Terrier
- !ruby/object:Mascota
  edad: "5"
  nombre: Ares
  raza: Labrador
```

FIN