

Wikidot.com

.wikidot.com

Share on      

[Edit](#) [History](#) [Tags](#) [Source](#)

[Explore »](#)

Ruby Tutorial

...o como pasar un buen rato programando

- [admin](#)
 - [site manager](#)

[Create account](#) or [Sign in](#)



Lección 1

- [Introducción](#)
- [Instalación](#)
- [El Primer Programa](#)
- [Números en Ruby](#)
- [Strings y diversión](#)
- [Variables](#)
- [Alcance de las variables](#)

Lección 2

- [Introduciendo Datos](#)
- [Normas en los nombres](#)

- [Los métodos](#)
- [Los métodos: argumentos](#)
- [Rangos](#)
- [Arrays](#)

Lección 3

- [Bloques](#)
- [Más malabares con strings](#)
- [Expresiones Regulares](#)
- [Condicionales](#)
- [Bucles](#)
- [Números Aleatorios](#)

Lección 4

- [Clases y Objetos](#)
- [Accesores](#)
- [Ficheros: lectura/escritura](#)
- [Cargando librerías](#)
- [Herencia de clases](#)
- [Modificando clases](#)
- [Congelando objetos](#)
- [Serializando objetos](#)

Lección 5

- [Control de acceso](#)
- [Excepciones](#)
- [Módulos](#)
- [Constantes](#)
- [Hashes y Símbolos](#)
- [La clase Time](#)

Lección 6

- [self](#)
- [Duck Typing](#)
- [Azúcar Sintáctico](#)
- [Test de unidades](#)

contacto

[e-mail](#)



Serializando Objetos

Java es capaz de **serializar objetos**: puede almacenarlos, para luego reusarlos cuando sea necesario. Ruby tiene también esta capacidad, pero la **llama `**marshaling`**.

Veamos un ejemplo en el que a partir de una clase, creamos una serie de objetos que almacenamos y luego recuperamos:

- La clase

personaje.rb

```
class Personaje
  def initialize(vida, tipo, armas)
    @vida = vida
    @tipo = tipo
  end
end
```

```

    @armas = armas
  end
  attr_reader :vida, :tipo, :armas
end

```

- Creamos los objetos y los guardamos en un fichero usando **Marshal.dump**:

```

require 'personaje'
p1 = Personaje.new(120, 'Mago', ['hechizos', 'invisibilidad'])
puts p1.vida.to_s+' '+ p1.tipo+' '
p1.armas.each do |w|
  puts w + ' '
end

File.open('juego', 'w+') do |f|
  Marshal.dump(p1, f)
end

```

- Usamos **Marshal.load** para recuperarlos:

```

require 'personaje'
File.open('juego') do |f|
  @p1 = Marshal.load(f)
end

puts @p1.vida.to_s + ' ' + @p1.tipo + ' '
@p1.armas.each do |w|
  puts w + ' '
end

```

Marshal únicamente **serializa** estructuras de datos; no puede serializar código (como hacen los objetos Proc), o recursos utilizados por otros procesos (como conexiones a bases de datos). Marshal da un error cuando se intenta serializar un fichero.

page_revision: 3, last_edited: 29 Jan 2009, 18:48 GMT-06 (669 days ago)

[EditTags](#) [History](#) [Files](#) [Print](#) [Site tools+](#) [Options](#)

[Help](#) | [Terms of Service](#) | [Privacy](#) | [Report a bug](#) | [Flag as objectionable](#)

Powered by [Wikidot.com](#)

Unless otherwise stated, the content of this page is licensed under [Creative Commons Attribution-ShareAlike 3.0 License](#)

Other interesting sites



Type Set Coin Collecting



zeromq



Scion: Monsters and Mosh Pits



[Wiki for 'My Vineyard'](#)

A Facebook Application