Desarrollo de aplicaciones web con Ruby on Rails

CISLA - Mayo 2013

Rayco Abad Martín
<*rayco@osl.ull.es*>

Oficina de Software Libre de la Universidad de La Laguna
http://osl.ull.es

Objetivos y Evaluación

Objetivo

- Adquisición de conocimientos básicos en el desarrollo de aplicaciones web con Ruby on Rails

Evaluación

- Asistencia de al menos el 80% (8 horas presenciales)
- Realización de actividades presenciales
- Trabajo Final

  Desarrollo de un conjunto de funcionalidades sobre la aplicación desarrollada durante el curso

Temario

- **Introducción a Ruby**
  - ¿Qué es Ruby?
  - Ruby Version Manager
  - Programación en Ruby
- **Ruby on Rails**
  - ¿Qué es Ruby on Rails?
  - Ruby Version Manager - GemSets
  - Programación en Ruby on Rails
    - Desarrollo guiado de un Blog

Introducción a Ruby

Introducción a Ruby
¿Qué es Ruby?

- Inspirado en Perl, Smalltalk, Eiffel, Ada y Lisp
- Programación Funcional
- Programación Imperativa
- Orientado a Objetos
- Multiplataforma
- Interpretado
- Tipado Dinámico
- Software Libre y Open Source (GPL y Ruby)
- ...

Introducción a Ruby
Ruby Version Manager

- Instalación de Ruby

```
$ sudo apt-get install ruby irb rdoc
```

Pero... ¿y si necesitamos diferentes versiones?

Permite la instalación y gestión de diferentes versiones de Ruby dentro de un mismo entorno

Introducción a Ruby
Ruby Version Manager (II)

- Instalación de RVM

```
$ sudo -s
# apt-get install curl
# curl -L get.rvm.io | bash -s stable
```

- Agregar usuarios al grupo "rvm" y reiniciar sesión

```
# usermod -G rvm -a <usuario>
```

- Para comenzar a usar RVM

```
# source /etc/profile.d/rvm.sh
```

Introducción a Ruby
Ruby Version Manager (III)

- Instalación de Dependencias

```
# rvm autolibs enable
# rvm requirements
```

- Instalación de Ruby desde RVM

```
# rvm install 1.9.3
```

- Versión por defecto

```
# rvm use 1.9.3 --default
```

Introducción a Ruby

- Números

```
>> 1 + 2                    #=> 3
>> 2 * 3                    #=> 6
>> 3 / 2                    #=> ¿...?
>> 3.0 / 2                  #=> ¿...?
>> 5 % 3                    #=> 2
```

Ejercicio:

¿Cuántos minutos tiene un mes de 30 días?

¿Cómo puedo calcular la raíz cuadrada de 9?

Introducción a Ruby

- Strings

```
>> 'Hola'                          #=> "Hola"
>> "Hola"                          #=> "Hola"
>> %q{Hola}                        #=> "Hola"
>> my_string = 'Mundo'             #=> "Mundo"
>> 'Hola' + my_string              #=> ¿...?
>> %Q{Hola #{my_string}}           #=> ¿...?
>> my_string.length                #=> 5
>> my_string * 3                   #=> ¿...?
>> my_string.include?('M')         #=> ¿...?
>> puts `ls`                       #=> ¿...?
```

Introducción a Ruby

- Las variables no necesitan ser declaradas (Duck Typing)

```
>> my_string = 'Hola Mundo'
```

  - Convenciones:

`variable` => Puede ser una variable local

`@variable` => Variable de objeto

`$variable` => Variable global

`VARIABLE` => Constante

- Comentarios

```
# Esto es un comentario
=begin
    Y ahora de varias líneas
=end
```

Introducción a Ruby

- Mi primer programa en Ruby

  ¿Hola Mundo?

```
>> puts("Hola Mundo")
>> puts 'Hola Mundo'
```

- Extensión **.rb** para los ficheros de código fuente

- Ejecutar con:

```
# ruby hola.rb
```

Introducción a Ruby

- Funciones

```
def say_hello(nombre)
    puts 'Hola ' + nombre
end
>> say_hello('Ruby')
>> say_hello 'Ruby'


def say_hello(nombre='Mundo')
    puts 'Hola ' + nombre
end
>> say_hello          #=> ¿...?
```

Introducción a Ruby

- Listas (Arrays)

```
>>[1, 2 , 3]                          #=> [1, 2, 3]
>> my_lista = [1, 2, 3]               #=> [1, 2, 3]
>> my_lista[0]                        #=> 1
>> my_lista[-1]                       #=> 3
>> letras = ['a', 'b']                #=> ["a", "b"]
>> letras = %w{a b}                   #=> ["a", "b"]
>> my_lista.each do |elem|
       puts elem                      #=> ¿...?
   end
```

Introducción a Ruby

- Hash

```
>> my_hash = {"a" => 1, "b" => 2} #=> {"b"=>2, "a"=>1}
>> my_hash['b']                   #=> 2
>> my_lista.keys                  #=> ["a", "b"]
>> my_lista.values                #=> ¿...?
>> my_lista['c'] = 3              #=> ¿...?
>> my_hash.keys.each do |key|
       puts my_hash[key]          #=> ¿...?
   end
```

Introducción a Ruby

- IF

```
if x > 10
    puts 'X es mayor que 10'
elsif x >= 5 && x <= 10
    puts 'X vale entre 5 y 10'
else
    puts 'X es menor que 5'
end
```

Introducción a Ruby

- CASE

```
case
    when x > 10 then puts 'X mayor que 10'
    when x >= 5 && x <= 10 then puts 'X entre 5 y 10'
    else puts 'X es menor que 5'
end
```

- UNLESS

```
unless x < 5
    puts 'X es mayor o igual que 5'
end
```

Introducción a Ruby

- WHILE

```
i = 0
while i < 10
    puts i
    i += 1
end
```

- UNTIL

```
i = 10
until i < 0
    puts i
    i -= 1
end
```

- FOR

```
for i in [0..9]
    puts i
end
```

- TIMES

```
10.times do |i|
    puts i
end
```

Introducción a Ruby

- CLASES

```ruby
class MiLibro
    def initialize(titulo, autor)      # Constructor
        @titulo = titulo               # Atributos
        @autor = autor
    end
    def titulo                         # Getter
        @titulo
    end
    def titulo=(mi_titulo)             # Setter
        @titulo = mi_titulo
    end
end
```

Introducción a Ruby

```
>> libro = MiLibro.new('Ruby', 'Matz' )
>> puts libro.titulo()
>> libro.titulo = 'The Ruby Programming Language'
>> libro.autor = 'Yukihiro Matsumoto'        #=> ERROR!
```

Ejercicio: Completa la clase MiLibro para poder mostrar y modificar el autor de un libro

Introducción a Ruby

SOLUCIÓN

```
class MiLibro
    attr_accessor :titulo, :autor          # Accesos
    def initialize(titulo, autor)          # Constructor
        @titulo = titulo                   # Atributos
        @autor = autor
    end
end
```

- Métodos de Acceso

```
attr_accesor :titulo # Acceso de lectura y escritura
attr_reader :titulo  # Acceso de lectura
attr_writer :titulo  # Acceso de escritura
```

Introducción a Ruby

- HERENCIA

```
class Animal
    def ladrar
        puts 'Guau'
    end
end

class Gato < Animal
    def maullar
        puts 'Miau'
    end
end
```

```
>> gato = Gato.new
>> gato.maullar
>> gato.ladrar
```

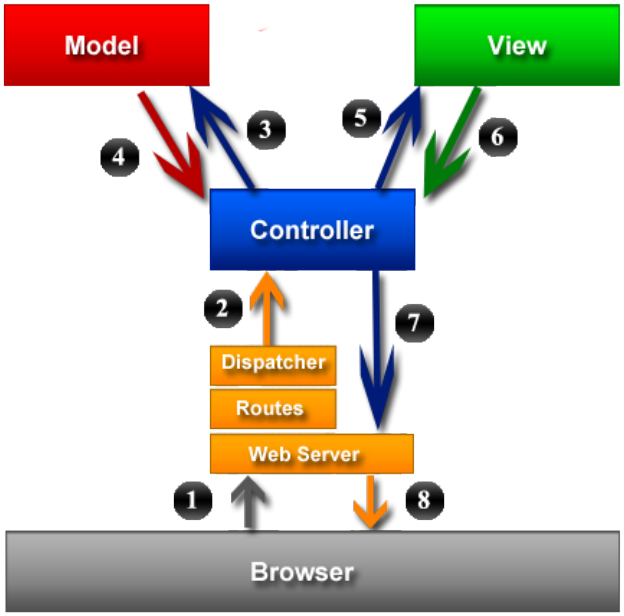
- SABER MÁS...

http://ruby-doc.org/
http://www.ruby-lang.org/es/
http://tryruby.org/
http://rubymonk.com/

Ruby on Rails

Ruby on Rails
¿Qué es Ruby on Rails?

- Framework Ruby para el desarrollo de aplicaciones Web
- ORM, REST, CRUD, DRY, ...
- Implementa el patrón de diseño de software MVC



Ruby on Rails
¿Qué es Ruby on Rails? (II)

MODELOS

Representa los datos de la aplicación y las reglas para manipularlos

Generalmente, cada tabla en la BBDD corresponde con un modelo

VISTAS

Representan la interfaz de usuario de la aplicación

HTML con código Ruby para la presentación de datos

CONTROLADORES

Procesan las solicitudes, interrogan a los modelos y pasan los datos a las vistas para su presentación

Ruby on Rails
GEMSETS

- RVM

  Permite gestionar distintas versiones de Ruby

  Permite definir gemsets

- GEMSETS

  Conjuntos de gemas que definimos para una versión de Ruby, aislados unos de otros gracias a RVM

DEFINIR UN GEMSET POR CADA PROYECTO

Ruby on Rails
GEMSETS (II)

- Creación

```
$ rvm gemset create <proyecto>
```

- Uso

```
$ rvm gemset use <proyecto>
```

También con

```
$ rvm gemset <versión de ruby>@<proyecto>
```

- Listar gemsets

```
$ rvm gemset list
```

```
$ rvm gemset help
```

Ruby on Rails
GEMSETS (III)

- ¿Tengo que cambiar de gemset al cambio de proyecto?
  1. Crear el fichero .rvmrc en la raíz del proyecto
  2. Editar el fichero y añadir el siguiente contenido:

     ```
     rvm --create use default@<proyecto> > /dev/null
     ```

  3. Guardar cambios

- Ventajas:
  - Carga automática del gemset de un proyecto
  - Si no existe el gemset, se crea.

Ruby on Rails
Primera Parte

Ruby on Rails

- Instalación de Ruby on Rails

```
$ gem install rails
```

- Crear esqueleto de la aplicación

```
$ rails new blog
```

- Ejecutar servidor de desarrollo

```
$ rails server
```

Nota: Una solución al tiempo de compilación de CoffeeScript de JS

```
$ sudo apt-get install nodejs
```

Ruby on Rails

- Configuración de la base de datos

  `./conf/database.yml`

  Entornos de desarrollo, test y producción

- Creación de la base de datos

```
$ rake db:create
```

- Acceso al SGBD desde Rails

```
$ rails dbconsole
```

Ruby on Rails

Ejercicio:

1. Crear el modelo `Post` que contenga los atributos `name` y `title` de tipo `string` y `content` de tipo `text`
2. Crear el controlador `Posts` con las funciones necesarias para CRUD

   Crear, Leer, Actualizar y Borrar

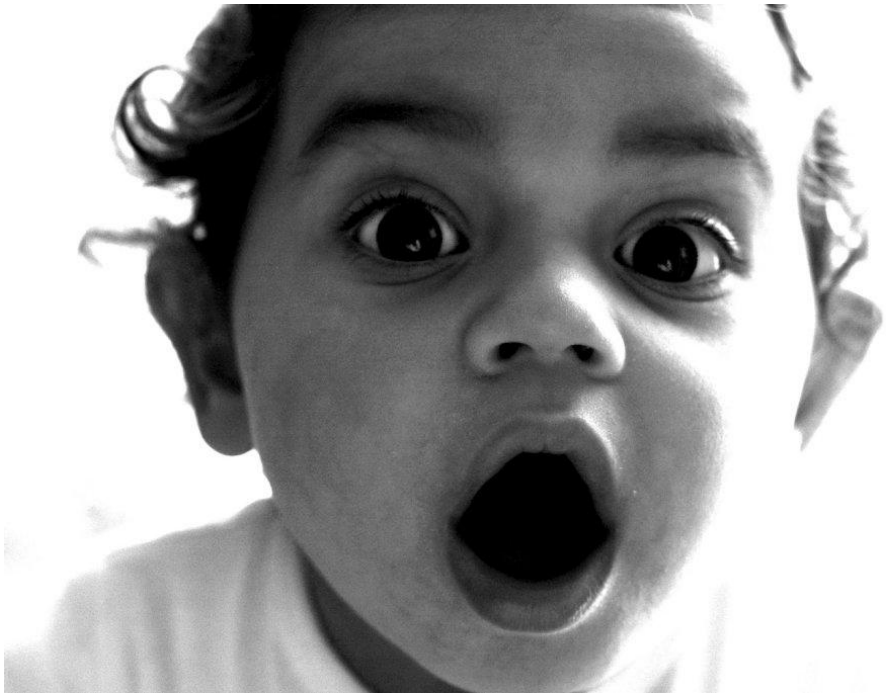
3. Creación de vistas asociadas a los Posts

Ruby on Rails

SOLUCIÓN:

```
$ rails generate scaffold Post name:string title:string
content:text
```

```
$ rake db:migrate
```



¿ASÍ DE FÁCIL?

Ruby on Rails
Segunda Parte

Ruby on Rails

¿Qué hemos visto hasta ahora?

- Creación de un gemset para el proyecto
- Instalación de Rails
- Creación de una nueva aplicación
- Estructura de la aplicación
- Creación del modelo Post con Scaffold
- Estudio del código autogenerado
- Layouts

Ruby on Rails

¿Qué nos queda por ver?

- Validación de formularios
- Migraciones
- Asociaciones
- Vistas Parciales
- Instalación de Gemas
- Autenticación de usuarios con Devise
- Trabajo Final

Ruby on Rails

- **VALIDACIÓN DE FORMULARIOS**

http://guides.rubyonrails.org/active_record_validations_callbacks.html

Ejercicio:

El campo `name` y el campo `title` deben estar presentes, este último además debe tener una longitud de al menos 5 caracteres

Cuando se cree un nuevo `Post`, si no se cumplen las validaciones se debe:

Recuadrar en rojo los campos erróneos del formulario

Mostrar un recuadro con todos los mensajes de error

Ruby on Rails

SOLUCIÓN:

Editar el modelo `Post` y añadir:

```
validates :name, :presence => true

validates :title, :presence => true,
                  :length => { :minimum => 5 }
```

¡Rails hace el resto!

Ruby on Rails

- **MIGRACIONES**

  http://guides.rubyonrails.org/migrations.html

  Ejercicio:

  Añadir a los `Posts` un campo `published` que indique si están publicados o no

  Editar la migración generada y añadir `false` como valor por defecto

  Modificar el modelo `Post` para darle acceso de escritura y lectura

  Modificar el formulario para poder indicar si el Post está publicado o no (`check_box`)

Ruby on Rails

SOLUCIÓN:

```
$ rails g migration AddPublishedToPosts published:boolean

$ rake db:migrate
```

Añadir valor por defecto:

```
class AddPublishedToPosts < ActiveRecord::Migration
    def change
        add_column :posts, :published, :boolean,  :default => false
    end
end
```

Añadir al formulario:

```
<div class="field">
    <%= f.label :published %>
    <%= f.check_box :published %>
</div>
```

Añadir `:published` a la lista de `attr_accessible`

Ruby on Rails

- **ASOCIACIONES**

http://guides.rubyonrails.org/association_basics.html

Ejercicio:

1. Generar el modelo `Comment` que contenga los atributos `commenter` de tipo `string` y `body` de tipo `text`, que hagan referencias a los `post`
2. Permitir que el modelo `Post` pueda tener muchos comentarios
3. Añadir las rutas para los comentarios pero que sean sólo accesibles desde los posts
4. Crear el controlador `Comments`
5. Crear un formulario para insertar comentarios desde un post

Ruby on Rails

SOLUCIÓN:

```
$ rails g model Comment commenter:string body:text
post:references
```

```
$ rake db:migrate
```

Editar el modelo `Post` y añadir:

```
    has_many :comments
```

Editar el fichero de rutas y asociar las rutas para los comentarios con las rutas de los posts

```
resources :posts do

  resources :comments

end
```

Ruby on Rails

```
$ rails g controller Comments
```

```
$ rake db:migrate
```

Editar la vista `show` de Post y crear el formulario

```
<h2>Add a comment:</h2>
<%= form_for([@post, @post.comments.build]) do |f| %>
  <div class="field">
    <%= f.label :commenter %><br />
    <%= f.text_field :commenter %>
  </div>
  <div class="field">
    <%= f.label :body %><br />
    <%= f.text_area :body %>
  </div>
  <div class="actions">
    <%= f.submit %>
  </div>
<% end %>
```

Ruby on Rails

¿PODEMOS CREAR YA COMENTARIOS?

No, necesitamos implementar la función `create` del controlador `Comments` que busque el post que se envía como parámetro y crearle el comentario

```
def create
  @post = Post.find(params[:post_id])
  @comment = @post.comments.create(params[:comment])
  redirect_to post_path(@post)
end
```

Ejercicio:

Además de mostrar los campos de un post, mostrar todos los comentarios asociados

Ruby on Rails

SOLUCIÓN:

Editar la vista `show` de `Post` y añadir:

```
<h2>Comments</h2>
<% @post.comments.each do |comment| %>
  <p>
    <b>Commenter:</b>
    <%= comment.commenter %>
  </p>
  <p>
    <b>Comment:</b>
    <%= comment.body %>
  </p>
<% end %>
```

Ruby on Rails

- **VISTAS PARCIALES**

  Permiten refactorizar el código de nuestras vistas, logrando códigos menos largos y complejos

  Convenio: El nombre de la vista debe comenzar con "_"

  Para renderizar una vista parcial basta con llamar a la función `render`

  ## Ejercicio:

  Crear la vista parcial `_comment.html.erb` con el código necesario para mostrar cada uno de los comentarios de un Post

  Renderizar el listado de todos los comentarios de un `Post` modificando para ello su vista `show`

Ruby on Rails

SOLUCIÓN:

Contenido de `_comment.html.erb`:

```
<p>
  <b>Commenter:</b>
  <%= comment.commenter %>
</p>
<p>
  <b>Comment:</b>
  <%= comment.body %>
</p>
```

Y renderizar desde la vista `show` de Post:

```
<%= render @post.comments %>
```

Ruby on Rails

¿PODEMOS BORRAR COMENTARIOS?

Ejercicio:

Implementar la función `destroy` del controlador `Comments` que busque el `Post` que se envía como parámetro (`:post_id`) y obtenga el comentario del Post que se quiere destruir (`:id`)

Una vez destruido se debe redireccionar al path del post

Añadir un enlace en nuestra vista parcial que permita eliminar el comentario

¿Qué hacemos si en lugar de borrar un comentario borramos un Post?

Ruby on Rails

SOLUCIÓN:

```
def destroy
  @post = Post.find(params[:post_id])
  @comment = @post.comments.find(params[:id])
  @comment.destroy
  redirect_to post_path(@post)
end
```

Añadir link para borrar el comentario desde la vista parcial:

```
<p>
    <%= link_to 'Destroy Comment', [comment.post, comment],
                :confirm => 'Are you sure?'
                :method => :delete %>
</p>
```

Modificar el modelo Post `has_many :comments, :dependent => :destroy`

Ruby on Rails

- **INSTALACIÓN DE GEMAS**

  RailRoady: Generador de diagramas de clase

  http://railroady.prestonlee.com/

  - Instalación

    ```
    $ sudo apt-get install graphviz
    ```

    Editar `Gemfile` y añadir `gem 'railroady'`

    Ejecutar `$ bundle install`

  - Uso

    ```
    $ rake diagram:all
    ```

Ruby on Rails

- **AUTENTICACIÓN DE USUARIOS CON DEVISE**

  https://github.com/plataformatec/devise

- Instalación

  Editar `Gemfile` y añadir `gem 'devise'`
  Ejecutar `$ bundle install`
  Ejecutar `$ rails g devise:install`

- Configuración

  Opción Host para el mailer

  Editar `config/environments/development.rb` y añadir:

```
config.action_mailer.default_url_options = { :host => 'localhost:
3000' }
```

Ruby on Rails

Añadir mensajes de devise en el layout

Editar `app/views/layouts/application.html.erb` y añadir:

```
<p class="notice"><%= notice %></p>

<p class="alert"><%= alert %></p>
```

- Creación modelo User

Ejecutar `$ rails g devise User`

Ejecutar `$ rake db:migrate`

- Rutas

Ejecutar `$ rake routes`

Ruby on Rails

- Interfaz de usuario

  Editar `app/views/layouts/application.html.erb` y añadir:

```
<div id="user_nav">
    <% if user_signed_in? %>
        Signed in as <%= current_user.email %>. Not you?
        <%= link_to "Sign out", destroy_user_session_path, :method => :delete
    %>
    <% else %>
        <%= link_to "Sign up", new_user_registration_path %> or
        <%= link_to "Sign in", new_user_session_path %>
    <% end %>
</div>
```

- Personalizar vistas

```
$ rails generate devise:views users
```

TRABAJO FINAL

Ruby on Rails
TRABAJO FINAL

FECHA LÍMITE DE ENTREGA: Domingo, 3 de junio de 2013

- Añadir un editor WYSIWYG (TinyMCE for Rails, ...)
- Añadir paginación al listado de Posts (will_paginate, kaminari, ...)
- Mejora de la interfaz de usuario (Twitter Bootstrap, ...)
- Modificar los formularios usando SimpleForm
- Asociaciones:
  - Un usuario tiene muchos posts
  - Un usuario tiene muchos comentarios
  - Un comentario es de un usuario
- **SABER MÁS...**

  http://guides.rubyonrails.org

  http://railscasts.com/

  http://railsforzombies.org/

  http://www.codeschool.com/paths/ruby

Desarrollo de aplicaciones web con Ruby on Rails

CISLA - Mayo 2013

¡GRACIAS!

Rayco Abad Martín

rayco.abad@gmail.com

https://github.com/rayco

http://www.linkedin.com/in/rabad