

Guía 5: Incorporación de código Ruby dentro del framework Rails.

Introducción

Ruby on Rails también conocido como RoR, es un framework escrito en lenguaje de programación Ruby desarrollado por David Heinemeier.

En esta guía se creará un pequeño proyecto haciendo uso del framework y se trabajará con las vistas de los usuarios finales, introduciendo código Ruby dentro estas, se aprenderá un poco sobre las hojas de estilos, como agregar un controlador nuevo a un proyecto, así como también se hará uso de archivos de configuración ya anteriormente utilizados.

Objetivos:

- Integrar código Ruby dentro de las vistas.
- Agregar estilos (CSS) a un proyecto en Rails.
- Crear controladores propios.

Tiempo

- Una sesión de clase.

Requerimientos

Software	Hardware
Sistema Debian 9 virtualizado en Virtual box con: <ul style="list-style-type: none">• Ruby versión 2.4.1• Rails 5.4.1• Nodejs• Sublimetext	Computadora con características: <ul style="list-style-type: none">• Memoria ram mínimo 2GB• Procesador mínimo 2.1 GHz

Referencia

- Michael Hartl. Ruby on Rails tutorial (Rail5), Learn Web Development with Rails.
<https://www.railstutorial.org/book/>
- LibrosWeb.es. capítulo 4. Hola, Rails!
http://librosweb.es/libro/introduccion_rails/capitulo_4.html
- Ruby on Rails org, RailsGuide, The Rails command line.
http://guides.rubyonrails.org/command_line.html

Desarrollo

1. Creación de un proyecto nuevo.

1.1. Ubicarse en el directorio donde se va almacenar el proyecto.

```
$ cd /home/debian/Proyectos_RoR
```

1.2. Generar un nuevo proyecto.

```
$ rails new my_app
```

1.3. Ubicarse dentro del proyecto e iniciar el servidor para comprobar que funciona sin ningún problema.

```
debian@debian:~/Proyectos_RoR$ cd my_app/  
debian@debian:~/Proyectos_RoR/my_app$ rails s  
=> Booting Puma  
=> Rails 5.1.4 application starting in development  
=> Run `rails server -h` for more startup options  
Puma starting in single mode...  
* Version 3.11.2 (ruby 2.4.1-p111), codename: Love Song  
* Min threads: 5, max threads: 5  
* Environment: development  
* Listening on tcp://0.0.0.0:3000  
Use Ctrl-C to stop
```

Figura 1 Mensajes al iniciar el servidor de Rails

1.4. Abrir el navegador y escribir la siguiente dirección.

```
http://localhost:3000
```

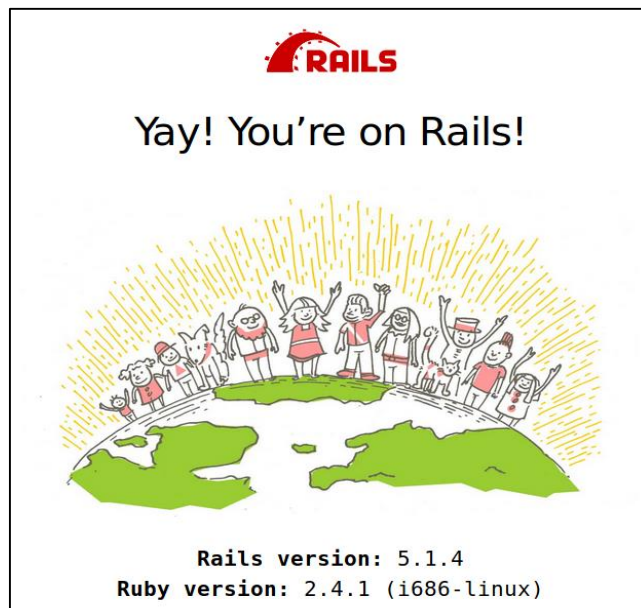


Figura 2 Index por defecto del proyecto

1. Generar un nuevo controlador al proyecto.

1.1. Verificar cuál es el controlador existente dentro del proyecto. De la siguiente forma se obtendrá el nombre de los controladores que existen hasta el momento.

```
$ ls app/controllers/*_controller.rb
```

Er

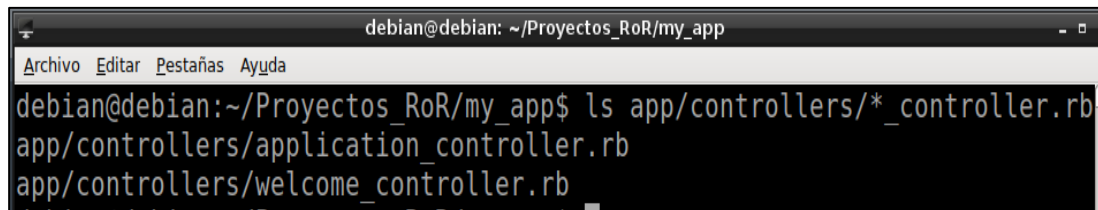
1.2. Generar el nuevo controlador y una acción correspondiente a ese controlador.

```
$ rails generate controller welcome index
```

```
Running via Spring preloader in process 20793
create  app/controllers/welcome_controller.rb
route   get 'welcome/index'
invoke  erb
create  app/views/welcome
create  app/views/welcome/index.html.erb
invoke  test_unit
create  test/controllers/welcome_controller_test.rb
invoke  helper
create  app/helpers/welcome_helper.rb
invoke  test_unit
invoke  assets
invoke  coffee
create  app/assets/javascripts/welcome.coffee
invoke  scss
create  app/assets/stylesheets/welcome.scss
```

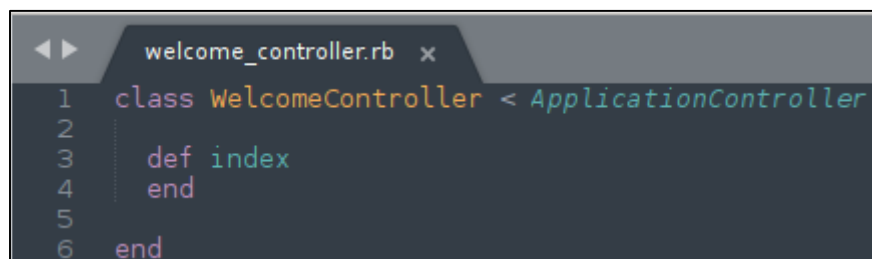
Con el comando anterior se genera y agrega un nuevo controlador (llamado welcome) al proyecto, de igual forma se crean un conjunto de archivos y directorios propios del controlador, que trabajan en conjunto para su funcionamiento, dentro de los más importantes se puede mencionar el archivo **welcome_controller.rb** y el archivo de las vistas llamado **index.html.erb**, que se pasó como parámetro al momento de generar el nuevo controlador. Dentro del controlador se crea una acción de nombre index que hace referencia a la vista index.html.erb del controlador, en la mayoría de los casos el controlador necesita de una vista por medio de la cual se muestra al usuario cada una de las acciones que puede realizar un controlador en específico.

- 1.3. Verificar nuevamente los controladores del proyecto. Podrá observar que ahora ya existe uno nuevo, y que se corresponde con el creado anteriormente.



```
debian@debian: ~/Proyectos_RoR/my_app
Archivo Editar Pestañas Ayuda
debian@debian:~/Proyectos_RoR/my_app$ ls app/controllers/*_controller.rb
app/controllers/application_controller.rb
app/controllers/welcome_controller.rb
```

- 1.4. **Abrir** el archivo **welcome_controller.rb** que se encuentra en el directorio (**app/controllers/**) y verificar que existe el método **index**, que fue creado al momento de generar el controlador, este es el método (acción) que tiene relación con la vista **index.html.erb**.



```
welcome_controller.rb x
1 class WelcomeController < ApplicationController
2
3   def index
4   end
5
6 end
```

Figura 3 Archivo del controlador

2. Trabajar con la vista

Para comenzar a trabajar con las vistas se debe ubicar el directorio (**app/view/**), que es donde se almacenan las vistas del proyecto, ahí se observa un archivo llamado **index.html.erb**, este es por medio del cual los usuarios finales interactúan con la aplicación

- 2.1. Editar el archivo **index.html.erb** del nuevo controlador y escribir el siguiente código html.

```
<h1>Bienvenido</h1>
<p>Página de inicio</p>
<p>UNAN-León</p>
```

- 2.2. Editar el archivo **welcome.scss**, que se encuentra en el directorio (**app/assets/stylesheets/**), en este directorio es donde se almacenan todas las hojas de estilos que contiene el proyecto. Todos los archivos de estilos en rails terminan en “.scss” debido a que el framework permite utilizar el lenguaje de estilos llamado Sass (Syntactically Awesome StyleSheets), es una extensión de css que agrega potencia y elegancia al lenguaje básico. Para darle un poco de formato a las vistas, agregar el siguiente código.

```
h1
{
  color: black;
  height: 30px;
  text-align: center;
}

p
{
  height: 15px;
  text-align: center;
}
```

- 2.3. Ir a la dirección (<http://localhost:3000/welcome/index>), podrá obtener una vista como la de la **figura 52**.



Como se observa en el navegador, se le ha aplicado formato al texto en la vista a través de las hojas estilos.

3. Si navega directamente en **http://localhost:3000/** no se tendrá acceso a la vista antes modificada, para esto, se debe modificar el archivo **routes.rb**, de manera que inicie (root) con la vista de nombre index.html.erb.



4. Actualizar el navegador en la dirección **http://localhost:3000/**

Al momento de actualizar el navegador ya se ha obtenido el acceso a la vista en la cual se ha estado trabajando a como se puede observar en la figura , esto es debido a que se ha especificado la ruta del controlador, como la ruta raíz o principal del proyecto.

5. Uso del lenguaje Ruby en el framework Rails.

Dentro del framework se trabaja con lenguaje Ruby; de igual forma en las vistas se puede agregar código Ruby a través del **lenguaje Embedded Ruby**, conocido como eRuby o ERB, haciendo uso de etiquetas propias del lenguaje ERB.

- 5.1. Para ver el funcionamiento ir al archivo **welcome_controller.rb** y crear un array global con los días de la semana dentro del **método index**.

```
@dias_semana =  
["Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo"]
```

- 5.2. Para poder hacer uso del array dentro de la vista, se debe utilizar lenguaje **ERB** en el archivo de la vista index.html.erb; editar el archivo y escribir el siguiente.

```
<h3>Imprimiendo un array Ruby dentro de una vista</h3>  
<% @dias_semana.each do |dia| %>  
  <%= dia %> <br>  
<% end %>
```

El código agregado a la vista hace uso de etiquetas propias del lenguaje ERB, como `<%%>` y `<%= %>`, que se utilizan para poder ingresar código Ruby dentro de las vistas, una de las diferencias entre estas 2 etiquetas es que esta (`<%= %>`) se utiliza cuando se imprime en la vista, como en el caso anterior que imprime cada uno de los datos almacenados en el array, y la etiqueta (`<% %>`) se utiliza para ejecutar código Ruby dentro las vistas pero sin mostrar ningún contenido, el uso de estas se puede observar en la siguiente figura.



- 5.3. De igual forma se puede hacer uso de los distintos métodos propios de los array, en el caso anterior se utilizó el método `each` para recorrer el array. Escribir el siguiente código en el archivo `index.html.erb`.

```
<h3> Método last</h3>
<%= @dias_semana.last %>
```

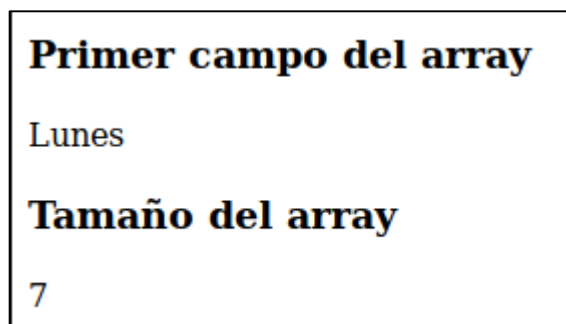
Utilizando el método **last**, se muestra en la vista el último dato almacenado en el array, a como se observa en la **figura 55**.



Ejercicios propuestos para ser entregados al docente

1. Realizar cada uno de los enunciados de la guía.
2. Muestre en la vista **index.html.erb** el primer dato almacenado y el tamaño completo del array, haga uso de métodos similares a los vistos anteriormente.

Ejemplo:



3. Generar un nuevo controlador de nombre “hash” y una acción que es la que tendrá relación con la vista llamada “mostrar”, similar al realizado al inicio de esta guía.
4. Dentro de la acción mostrar que se creó junto al nuevo controlador del proyecto, crear un hash con los datos de una persona: nombre, apellido, teléfono, correo.
5. Modificar la vista del controlador creado anteriormente y mostrar los datos del hash en una tabla html.
6. Agregar código css a la tabla, se mostrará en la vista como en la figura que se puede ver que la tabla contiene formato, colores, bordes.
7. Configurar el archivo **routes.rb** para cambiar la página de inicio del proyecto y ahora hacerlo apuntar al nuevo controlador junto con su respectiva vista, la cual deberá quedar como se observa en la figura

