# 

TEMAS:

* Ruby on Rails
  + Instalar Rails
  + bundle & gem
  + Estructura de Archivos
  + Entornos de Desarrollo
  + MVC
  + Controladores, rutas y vistas
  + Scaffold
  + Métodos de petición HTTP

OBJETIVO:

Aprender qué es Rails y su funcionamiento básico.

MARCO TEÓRICO

## Lectura 1: RAILS

**¿Por Qué Rails?**

Rails es un framework de desarrollo de aplicaciones web escrito en el lenguaje de Ruby. Está diseñado para simplificar la programación de aplicaciones web, asumiendo que todos los desarrolladores necesitan empezar. Este diseño hace que se tenga que escribir menos código que en otros frameworks. Experimentados desarrolladores Rails dicen que el desarrollo de la aplicaciones web se vuelve mucho más divertido.

Rails es un software obstinado. Esto hace suponer que existe una mejor manera de hacer las cosas, y está diseñado para alentar este camino -y en algunos casos desalentar alternativas. Si tu aprendes el "Camino Rails" probablemente descubrirás un tremendo incremento en la productividad. Si tu persistes en traer viejos hábitos de otros lenguajes a tu desarrollo Rails, tratando de utilizar patrones aprendidos en otros sitios, es posible que tengas una experiencia menos feliz.

La filosofía Rails incluye dos principios rectores:

* No te repitas a ti mismo (Don't Repeat Yourself): DRY es un principio del desarrollo de software el cual establece "Cada pieza del conocimiento debe ser única, sin ambigüedades, con su propia autoridad y representación dentro de un sistema". Al no escribir la misma información una y otra vez, nuestro código es más fácil de mantener, más reutilizable y con menos errores.
* Convención sobre Configuración: Rails tiene sus opiniones acerca de el mejor de hacer muchas cosas en las aplicaciones web, y por defecto las configura como convenciones, en lugar de requerir que se especifiquen minuciosamente a través de ficheros de configuración sin fin.

## Creando un nuevo Proyecto Rails

El mejor camino para utilizar esta guía es seguir cada paso como ocurre, ningún código o paso se dejará afuera para que puedas seguir literalmente paso por paso hasta el final.

Para seguir hasta el final este capítulo, crearás un proyecto Rails llamado blog, un (muy) sencillo web\_blog. Antes de que puedas empezar a construir una aplicación, necesitas estar seguro de que tienes instalado rails.

**Instalando Rails**

Abre una terminal de línea de comandos. En Mac OS X abre Terminal.app, en Windows elige "Ejecutar" desde tu menú de Inicio y escribe 'cmd.exe'. Cualquier comando precedido de un signo $podría ser ejecutado en la línea de comandos. Verifica que tienes una versión actualizada de Ruby instalada:

$ ruby -v

ruby 2.0.0p353

En muchos sistemas operativos tipo UNIX encuentras una aceptable version de SQLite3. En Windows, si tu estás instalando Rails a través del Instalador de Rails, ya tendrás SQLite instalado. Otros pueden encontrar instrucciones de instalación en la Página de SQLite3. Verifica que está correctamente instalada en tu PATH:

|  |
| --- |
| $ sqlite3 --version |

El programa mostrará su versión.

Para instalar Rails, utiliza el comando gem install proporcionado por RubyGems:

|  |
| --- |
| $ gem install rails |

Para verificar que tienes todo instalado correctamente, deberías tener la posibilidad de ejecutar el siguiente comando:

|  |
| --- |
| $ rails --version |

Si dice algo como "Rails 5.0.0", estás listo para continuar.

#### **Creando la Aplicación Blog**

Rails viene con un número de scripts llamados generadores (generators) que están diseñados para hacer tu vida de desarrollador más fácil creando todo lo que necesitas para empezar a trabajar en una tarea particular. Una de esos es el generador de nuevas aplicaciones Rails, el cual provee lo necesario para crear una aplicación rails fresca, y que no tengas que hacerlo manualmente tu mismo.

Para utilizar este generador, abre la terminal de línea de comandos, navega hasta el directorio donde tengas permisos para crear ficheros, y escribe:

|  |
| --- |
| $ rails new blog |

Esto creará una aplicación Rails llamada Blog en un directorio blog e instalará las gemas de las que dependa enumeradas en el Gemfile utilizando bundle install.

Después de crear la aplicación blog, entra en el directorio:

|  |
| --- |
| $ cd blog |

El directorio blog tiene un número de ficheros y carpetas auto-generados que constituyen la estructura de una aplicación Rails. La mayor parte del trabajo en este tutorial se hará en la carpeta app, pero aquí está la relación básica entre cada fichero/carpeta y la función que cumple:

|  |  |
| --- | --- |
| **Archivo/Directorio** | **Propósito** |
| app/ | El centro de tu aplicación , contiene los controllers, models, views, helpers, mailers y assets para tu aplicación. |
| app/assets | Los assets de la aplicación tales como hojas de estilos, javaScript y imágenes. |
| bin/ | Los ejecutables en binario |
| config/ | Configura las reglas de ejecución de la aplicación, rutas, base de datos y más. |
| db/ | Contiene el esquema actual de tu base de datos, así como las migraciones de la base de datos. |
| doc/ | Documentación detallada de la aplicación |
| lib/ | Módulos extendidos para tu aplicación. |
| lib/assets | Librerias externas de assets (CSS, JS, imagenes) para la aplicación. |
| log/ | Los Logs de la aplicación. |
| public/ | La única carpeta vista por el mundo tal como es. Contiene los archivos estáticos y assets compilados. |
| bin/rails | Scripts para generar código, abrir sesiones de consola o iniciar el servidor. |
| test/ | Archivos para pruebas. |
| tmp/ | Archivos temporales. |
| vendor/ | Lugar para código de terceros. En una típica aplicación Rails, ésta incluye librerías y plugins. |
| vendor/assets | Lugar para assets de terceros, plugings, librerias. |
| README.md | A brief description of the application |
| Rakefile | Este archivo localiza y carga tareas que pueden ser ejecutadas desde la línea de comandos. |
| Gemfile | La lista de las gemas requeridas por la palicación. |
| Gemfile.lock | Ula lista de las gemas usadas por la aplicación, especificando la versión usada y las dependencias de cada una. |
| config.ru | Configuración Rack para servidores basados en Rack usados para iniciar la aplicación. |
| .gitignore | Archivos a ignorar en el momento de usar Git |

Bundler

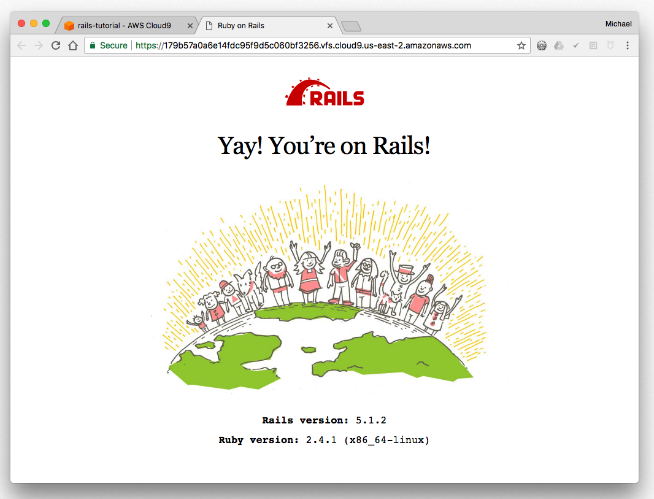
Bundle es una gema que nos ayudará con el manejo las gemas necesarias en nuestra aplicación. Normalmente después de crear una nueva aplicación en rails, el siguiente paso es usar Bundler (bundle install) este comando se ejecuta automático cuando ejecutamos el generador de aplicaciones que nos provee rails (rails new) pero cada vez que nosotros hacemos un cambio referente a las gemas de nuestra aplicación, es necesario volver a ejecutarlo.

#### **Levantando el Servidor Web**

Realmente, ya tienes una aplicación Rails funcional. Para ver como funciona, solo necesitas levantar el servidor web en tu ordenador de desarrollo. Puedes hacer esto ejecutando lo siguiente en el directorio blog:

|  |
| --- |
| $ bin/rails server |

Esto levantará WEBrick, un servidor web distribuido con ruby por defecto. Para ver tu aplicación en acción, abre una ventana de navegador en la siguiente dirección [http://localhost:3000](http://localhost:3000/). Deberías ver la página de información por defecto de Rails:



La página de "Bienvenido a bordo" es una prueba para una nueva aplicación Rails: asegura que tienes configurado el software lo suficientemente bien para servir una página.

MVC (modelo - vista - controlador)

El patrón de arquitectura modelo vista controlador (MVC) impone la separación de entre los datos de la aplicación (tales como la información del usuario) y el código utilizado para mostrarlo, que es una forma común de estructurar una interfaz gráfica de usuario, puedes notar en la estructura de la aplicación estándar de Rails, dentro del directorio app contiene tres subdirectorios model, views, controller.

Cuando interactúa con Rails el Browser envía una solicitud que es recibida por el servidor web y enviada al controlador. En algunos casos, el controlador mostrará inmediatamente una vista, que es una plantilla que se convierte a HTML y se envía de vuelta al navegador. Más comúnmente para sitios dinámicos, el controlador interactúa con un modelo, que es un objeto Ruby que representa un elemento del sitio (como un usuario) y se encarga de comunicarse con la base de datos. Después de invocar el modelo, el controlador muestra la vista y devuelve la página web completa al navegador como HTML.

#### 

#### **Decir "Hola", Rails**

Para obtener que Rails diga "Hola", necesitas crear como mínimo un *controlador* y una *vista*.

El propósito de un controlador es recibir una petición específica para la aplicación. *El sistema de enrutamiento* , decide qué controlador recibe cada petición. A menudo, hay más de una ruta hacia cada controlador, y las diferentes rutas pueden ser servidas por diferentes *acciones*. El propósito de cada acción es recoger información para proveerla a una vista.

El propósito de una vista es mostrar esa información en un formato humano legible. Una importante distinción para hacer es que es en el *controlador*, no en la vista, donde la información se recoge. La vista debe sólo mostrar esa información. Por defecto, las vistas están escritas en un lenguaje llamado eRuby (Ruby Embebido), el cual es procesado en el ciclo de la petición en Rails antes de ser enviada al usuario.

Para crear un nuevo controlador, necesitarás ejecutar el generador "controller" y decirle que quieres un controlador llamado "bienvenido" con una acción llamada "index", de la siguiente manera:

|  |
| --- |
| $ bin/rails generate controller bienvenido index |

Rails creará varios ficheros y rutas por ti.

|  |
| --- |
| create app/controllers/bienvenido\_controller.rb  route get 'bienvenido/index'  invoke erb  create app/views/bienvenido  create app/views/bienvenido/index.html.erb  invoke test\_unit  create test/controllers/bienvenido\_controller\_test.rb  invoke helper  create app/helpers/bienvenido\_helper.rb  invoke assets  invoke coffee  create app/assets/javascripts/bienvenido.coffee  invoke scss  create app/assets/stylesheets/bienvenido.scss |

Lo más importante de esto es por supuesto el controlador, localizado en app/controllers/bienvenido\_controller.rb y la vista, localizada en

app/views/bienvenido/index.html.erb.

Abre el fichero app/views/bienvenido/index.html.erb con tu editor de texto. Borra todo el código existente en el fichero, y reemplazalo con la siguiente linea de código:

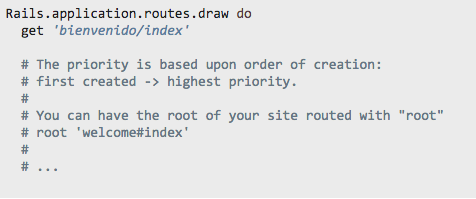
<h1>¡Hola, Rails!</h1>

#### **Configurando la página de Inicio de la aplicación**

Ahora que tenemos hecho el controlador y la vista, necesitamos decirle a Rails cuando queremos que muestre "¡Hola, Rails!". En nuestro caso queremos que lo haga cuando el navegante pida la raíz de nuestro sitio, es decir esta URL, [http://localhost:3000](http://localhost:3000/). Hasta el momento, "Bienvenido a bordo" está ocupando ese lugar.

Para continuar, tienes que decirle a Rails donde tu página de inicio actual se localiza.

Abre el fichero config/routes.rb en tu editor.



Esta es el fichero de *enrutamiento* de tu aplicación el cual mantiene las entradas en un lenguaje especial DSL (domain-specific language) que dice a Rails como conectar las peticiones entrantes con los controladores y las acciones. Este fichero contiene muchas líneas comentadas que son rutas rutas de ejemplo, y una de ellas actualmente muestra como conectar a la raíz de tu sitio con una acción y controlador específicos. Encuentra la línea que comienza con root, descoméntala y reemplaza welcome por bienvenido. Debería verse algo como esto:

root 'bienvenido#index'

root 'bienvenido#index' le dice a Rails que derive las peticiones que llegan a la raíz de la aplicación hacia la acción index del controlador bienvenido y get 'bienvenido/index' le indica a Rails que las peticiones que llegan a http://localhost:3000/bienvenido/index se deriven a la acción index del controlador bienvenido también. Esto fue creado automáticamente cuando ejectutasteis por comando el controller generator (rails generate controller bienvenido index).

Levanta nuevamente el servidor web si lo habías parado para generar el controlador (rails server) y abre el navegador en http://localhost:3000. Verás el mensaje "¡Hola, Rails!" que has escrito en el fichero app/views/bienvenido/index.html.erb, indicando que la nueva ruta en efecto va a la acción index del BienvenidoController y muestra la vista correctamente.

## Empezar y Ejecutar

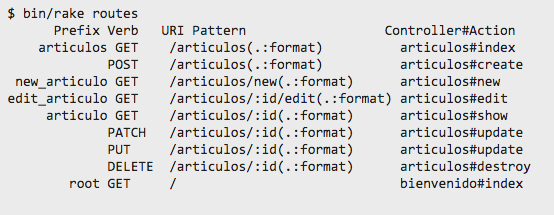
Ahora que has vistos como se crea un controlador, una acción y una vista, vamos a crear un poco más ambicioso.

En la aplicación Blog, crearás un nuevo *recurso*. Un recurso es el termino utilizado para una colección de objetos similares, como artículos, gente o animales. Puedes crear, leer, actualizar y destruir elementos de un recurso, estas operaciones son conocidas como operaciones *CRUD*.

Rails provee un método para recursos que puedes utilizarlo para declarar un recurso REST. Necesitas añadir el *recurso articulo* al fichero de configuración del enrutamiento config/routes.rb de la siguiente manera:



Si ejecutar rake routes, verás que como se han definido las rutas para todas las acciones RESTful estándar. El significado de la columna prefijada (y las otras columnas) lo veremos más adelante, pero ahora nota como Rails ha deducido la forma singular articulo y hace un uso significativo de esta.



En la próxima sección, añadiremos la capacidad de crear nuevos artículos en tu aplicación y será posible verlos. Estas son la "C" y la "R" de "CRUD": crear y leer (creation and reading en inglés).

La primera cosa que necesitaremos para crear un nuevo artículo, es crear una sección especial para esto en nuestra aplicación. Un buen lugar podría ser /posts/new. Si tratas de navegar a esa dirección ahora - visitando http://localhost:3000/articles/new - Rails mostrará el siguiente mensaje de error:



Este error se produce porque la ruta necesita tener asociado un controlador para poder servir las peticiones. Así que la solución es tan sencilla como crear un controllador llamado ArticlesController. Si no quieres crearlo a mano, ejecuta el siguiente comando:

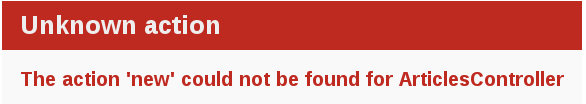


Si ahora abres el archivo app/controllers/articles\_controller.rb que se acaba de generar, verás el siguiente controlador vacío:

class ArticlesController < ApplicationController  
end

Un controlador es simplemente una clase que hereda de ApplicationController. Dentro de esta clase definen los métodos que se convertirán en las acciones del controlador. Estas acciones son las que se encargan de las operaciones CRUD sobre los artículos de la aplicación.

Si ahora refrescas la página http://localhost:3000/articles/new, verás de nuevo un error, pero diferente al anterior:



Este error significa que Rails no puede encontrar la acción new dentro del controlador ArticlesController que acabas de generar. El motivo es que cuando se genera un controlador automáticamente, su contenido está vacío a menos que le indiques qué acciones quieres incluir.

Para añadir una acción a mano en un controlador, lo único que necesitas es crear un nuevo método en el controlador. Abre el archivo app/controllers/articles\_controller.rb y dentro de la clase ArticlesController define un nuevo método de esta manera:

def new  
end

Refresca de nuevo la página http://localhost:3000/articles/new y verás que a pesar de haber añadido el método new, sigues viendo un mensaje de error:



La causa de este error es que Rails espera que las acciones de los controladores tengan asociadas unas vistas para mostrar su información.

Rails tratará de encontrar una plantilla llamada articles/new en el directorio app/views. El formato para esta plantilla sólo puede ser html y el manejador debería ser erb, builder o coffee. Ya que quieres crear un nuevo formulario HTML, tendrás que usar el lenguaje ERB. Por lo tanto el archivo debería llamarse articles/new.html.erb y debe encontrarse dentro de la carpeta app/views de la aplicación.

Ahora crea un nuevo archivo en app/views/articles/new.html.erb y escribe el siguiente contenido en él:

<h1>New Post</h1>

Cuando recargues http://localhost:3000/posts/new, verás que la página tiene un título. La ruta, el controlador, la acción y la vista ahora están funcionando todos de manera conjunta y por eso ya no se muestra ningún error. Ahora es el momento de crear un formulario para crear artículos.

Herramientas

Instalar rails ejecutable para Windows, fundamental que el equipo tenga instalado node.js

Ejercicios

* Generar un proyecto Rails.
* Adicione un landing-page de su proyecto a el framework.

