Ruby on Rails, una forma rápida de hacer aplicaciones web

Roberto Solano Murillo
Universidad de Costa Rica,
Escuela de Ciencias de la Computación e Informática,
San José, Costa Rica
rasm007@gmail.com

Eduardo Coles
Universidad de Costa Rica,
Escuela de Ciencias de la Computación e Informática,
San José, Costa Rica
klx.klx@gmail.com

Abstract

In this article we will explain what is *Ruby on Rails* (RoR) . How was it born and which was the purpose of its creators. We will explore in deep its main features, compared with other frameworks of web applications. Besides we will describe its functionality as well as its context use. Also the idea of this article is not only and introduction to the topic but also a small tutorial about the basics of RoR programming. So we will talk from since how to compile up to how to install RoR.

Keywords: web applications, framework, programming languages, ruby

Resumen

En este artículo explicaremos que es *Ruby on Rails* (RoR) . Cómo nació y cuál fue el propósito de sus creadores. Exploraremos a fondo cuáles son sus principales ventajas, respecto a otros *frameworks* de aplicaciones web. Asimismo describiremos su funcionalidad así como su contexto de uso. Aparte de todo lo anterior, la idea es que este artículo no solo sea una introducción al tema sino también una guía de cómo empezar a programar usando RoR. Así que hablaremos desde cómo conseguir el compilador hasta cómo instalarlo.

Palabras clave: aplicaciones web, framework, lenguajes de programación, ruby

1. Introducción

Con el auge de la Internet en los últimos tiempos, cada vez es más el desarrollo y la importancia de la implementación de aplicaciones web de forma fácil y rápida. Por eso últimamente ha tenido un gran auge RoR como *framework* de desarrollo de aplicaciones web (ver figura 1). Pero vamos a hablar un poco de sus orígenes. Ruby fue inventado por un japonés llamado Yukihiro Matsumoto. Ruby viene de Perl, y la idea era tener un lenguaje que fuera más poderoso que Perl y más orientado a objetos que Python. Es un lenguaje de scripts, multiplataforma, netamente orientado a objetos y lo mejor de todo es que es software libre. RoR se originó con una aplicación de administración de proyectos conocida como Basecamp desarrollada por el danés David Heinemeier Hansson para la compañía 37signals [4]. Originalmente David intentó escribir Rails en PHP pero fracasó, hasta que uso Ruby. RoR es un framework para el desarrollo de aplicaciones web, software libre y basado en el patrón de diseño Modelo Vista Controlador (MVC) [3].

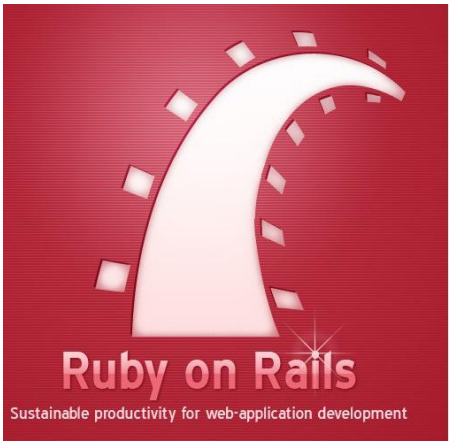

***Figura 1. Logo Ruby on Rails*.**

RoR nos da muchas ventajas si lo comparamos con otros *frameworks*. Una de las principales es que es gratis(*open source*) y otra que posee toda una comunidad de apoyo. Rails está basado en dos principios principales de desarrollo:

- Don't repeat yourself (DRY)
- Convención sobre configuración

Más adelante detallaremos más sobre que significan estas dos características.

RoR se basa en el desarrollo ágil y RUP (Proceso Unificado Racional ó *Rational Unified Process* en inglés) por lo que según el contexto no siempre puede ser la mejor opción de desarrollo. Dependiendo del material humano disponible así como las características del proyecto a desarrollar depende de que tan útil vaya a resultar como opción.

Empezar a usar RoR es muy simple, para empezar solo ocupamos descargar Ruby, Ruby Gem y Rails. Pero esto lo explicaremos más ampliamente en el desarrollo.

2. Framework

Los *frameworks* son parte fundamental en la ingeniería del software, ya que promueven la reutilización del código del diseño y el código fuente. Los puntos flexibles de un framework se llaman *los puntos calientes (hot-spots).* Los puntos calientes o *Hot-spots* son las clases o los métodos abstractos que deben ser implementados o puestos en ejecución. … Algunas de las características del framework no son mutables ni tampoco pueden ser alteradas fácilmente. Estos puntos inmutables constituyen el núcleo o kernel de un *framework*, también llamados como los puntos congelados o *frozen-spots* del framework. A diferencia de los puntos calientes o *hot-spots*, los puntos congelados o inmutables son los pedacitos del código puestos en ejecución ya dentro del framework que llaman a uno o más puntos calientes proporcionados por el ejecutor. El núcleo o *Kernel* será la constante y presentará siempre la parte de cada instancia del *framework* [2].

"Piense en un *framework* como si fuese un motor. Un motor requiere potencia. A diferencia de un motor tradicional, un motor del *framework* tiene muchas entradas de potencia. Cada uno de estas entradas de potencia es un punto caliente del *framework*. Cada punto caliente debe ser accionado para que el motor funcione. Los generadores de potencia son el código específico de la aplicación que se debe enchufar a los puntos calientes. El código agregado de la aplicación será utilizado por el código *kernel* del *framework*. El motor no correrá hasta que todos los enchufes esten conectados" [2]. (Ver figura 2).

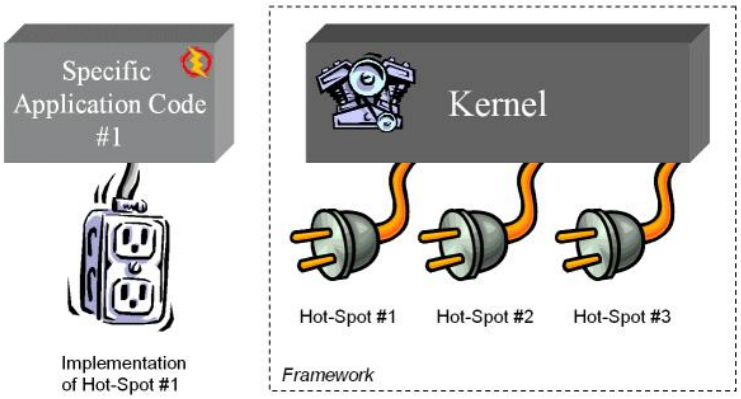

Figura 2. Framework.

2.1 Ventajas de RoR respecto a otros *frameworks*

Rails usa convenciones, paquetes de programación integrados y código predefinido, diseñado para completar y usar inmediatamente sin necesidad de configuración. A diferencia de otros ambientes de programación, como aquellos basados en Java que requieren usar varios frameworks, los cuales deben ser configurados para que funcionen entre sí, para obtener todas las capacidades deseadas [5].

Entre los fundamentos de RoR están los siguientes:

- DRY (*Don't Repeat Yourself*) : "no te repitas a ti mismo", con esto podemos tener un formulario, y llamarlo las veces que queramos y desde donde queramos, simplemente con una línea código, o tal vez tener una tabla en nuestra base de datos, y manipular a los registros como un objeto y a sus campos como un atributo, sin necesidad de que declaremos nada.
- Convención sobre configuración:

```
class Auto < ActiveRecord::Base
end
```

  Con esa declaración de una clase, mapeamos a una tabla en nuestra base de datos, dicho de otra manera Rails buscara una tabla llamada 'autos', en nuestra base de datos, y porque en plural?, esto es así porque Rails cree conveniente que debe llamarse así (principio de pluralización), aunque este comportamiento se puede desactivar de una manera muy sencilla, y si no la encuentra?, pues nos dará un error.
  Y si la tabla con la que quiero trabajar no tiene ese nombre exacto?, no hay problema, con una línea más lo podemos solucionar:

```
class Auto < ActiveRecord::Base
  set_table_name 'carros'
end
```

  Con esto el *framework* comprenderá que en vez de usar 'autos', debería usar 'carros' .
- Uso de patrones de diseño: Modelo Vista Controlador (MVC)
- Generación de código (*helpers*): permiten la generación de código xhtml, xml y javascript a partir de código Ruby.
- Menos código, menos errores.
- Test integrados (unitarios y funcionales).

Todas estas ventajas las explotaremos según el uso que vayamos a darles en un determinado proyecto.

3. Contexto de uso

3.1 Cuando NO usar Rails

- Con aplicaciones muy grandes
- Con bases de datos legadas
- Con desarrolladores poco habituados a los cambios
- Con equipos de desarrollo muy grandes

3.2 Cuando usar Rails

- Cuando se domina el *framework*
- Con equipos ágiles y dinámicos
- Cuando se tienen pocos desarrolladores
- Con proyectos pequeños o medianos

Con esto ya sabemos cuando usarlo, pero ahora necesitamos entender como funciona internamente RoR.

4. El patrón MVC

La aplicación se divide en tres partes:

1. Modelo: responsable de mantener los datos de la aplicación
2. Vista: se encarga de la interfaz y presentar la información al usuario
3. Controlador: hace cálculos y une todas las piezas. Recibe eventos del exterior, interactúa con el modelo y actualiza la información de las vistas.

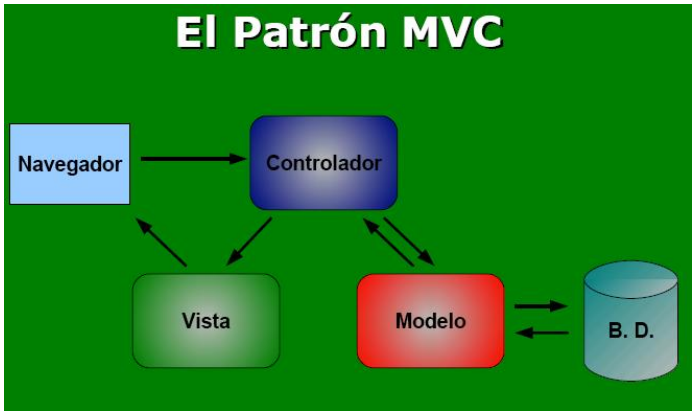

Figura 3. Diagrama del patrón MVC.

En la figura 3 podemos apreciar cómo se relacionan las diferentes etapas del patrón MVC. En la figura 4 vemos la arquitectura básica en RoR, que se compone de la siguiente manera.

El modelo (*Módulo: ActiveRecord*) : una clase representa una tabla, se descubren automáticamente los campos, se pueden declarar relaciones con otros modelos/tablas, se puede personalizar y añadir métodos.

La vista (*Módulo: ActionView*): HTML con Ruby embebido, una por cada acción de cada controlador, dependen de los controladores, plantillas en formato *Embedded Ruby*, y HTML repetitivo, a funciones externas (*helpers*).

El controlador (*Módulo: ActionPack*): cada controlador es una clase de Ruby, cada método es una acción, y tiene andamios (*scaffolds*) para avanzar más rápido.

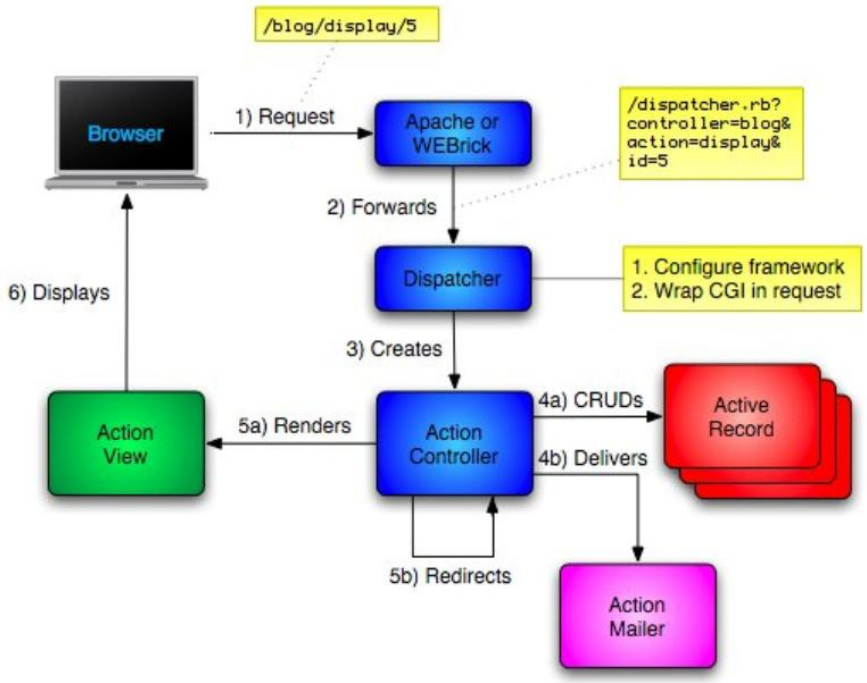

Figura 4. Arquitectura básica de MVC.

En la figura 5 podemos ver el proceso de la petición de una página en la arquitectura MVC. Primero, se envía la petición. Segundo, se traen los datos de la base de datos. Tercero, los datos son presentados a la interfaz. Y finalmente, los datos son enviados de vuelta al navegador.

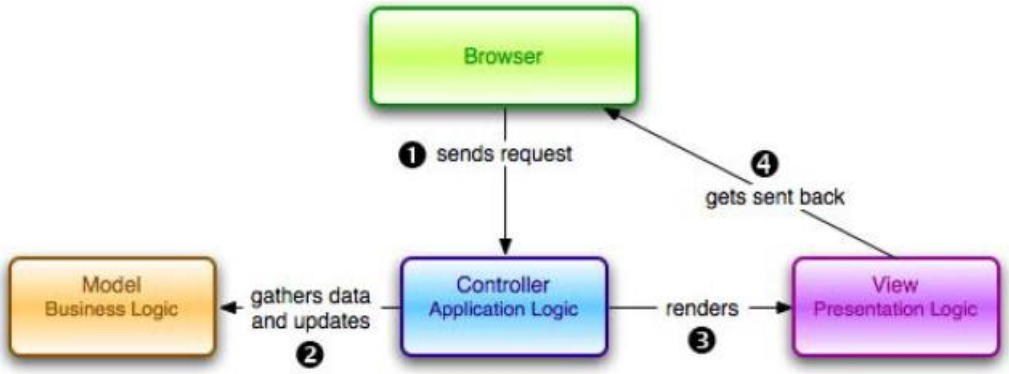

Figura 5. Procesando la petición de una página en la arquitectura MVC.

5. Requerimientos para usar RoR

Para empezar a usar RoR debemos descargarnos Ruby y Rails, por aparte. Los pasos a seguir obviamente dependerán de la plataforma en la que quieras usar Rails. Rails trabaja con varios servidores web y bases de datos. Para el servidor web, se recomienda Apache o lighttpd ejecutando FastCGI o SCGI. Para la base de datos, podemos usar MySQL, PostgreSQL, SQLite, Oracle, SQL Server, DB2, o Firebird. Lo bueno es que funcionará sobre cualquier sistema operativo, pero recomendamos alguno basado en Unix para el desarrollo [6]. Hablaremos de los casos más usados: Windows y Linux.

5.1 Instalando RoR en Windows

Para instalar RoR en Windows tenemos disponible:

- InstantRails: lo podemos bajar de acá http://instantrails.rubyforge.org/wiki/wiki.pl . La ventaja de *InstantRails* es que al bajarlo no necesitamos bajar Ruby y Rails por aparte, ya que viene todo listo de una vez. Tampoco debemos preocuparnos por instalarlo ya que al descargarlo viene listo para ejecutarlo lo cual es algo bastante cómodo.
- XAMPP + AxleGrease: XAMPP es un servidor independiente de plataforma, software libre, que consiste principalmente en la base de datos MySQL, el servidor web Apache y los interpretes para lenguajes de script: PHP y Perl. El nombre proviene del acronimo de **X** (para cualquiera de los diferentes sistemas operativos), **A**pache, **M**ySQL, **P**HP, **P**erl. El programa esta liberado bajo la licencia GNU y actua como un servidor web libre, fácil de usar y capaz de interpretar páginas dinámicas [7]. AxleGrease es una extensión para XAMPP que nos permitirá poder usar Rails. (Nota: esta opción es multiplataforma)

5.2 Instalando RoR en Linux

Para instalar RoR en Linux seguimos los siguientes pasos:

1. Instalando Ruby:
   Descargamos el fuente desde ftp://ftp.ruby-lang.org/pub/ruby/ruby-1.8.5.tar.gz y lo descomprimimos

   ```
   ./configure
   make
   make install
   ```

   Luego instalamos el gestor de paquetes Rubygems:
   descargamos desde http://rubyforge.org/frs/download.php/11289/rubygems-0.9.0.tgz , luego descomprimimos y:

   ```
   cd rubygems-0.9.0
   ruby setup.rb
   ```

   Para usuarios de Debian o Ubuntu

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

   y el mismo procedimiento para Rubygems.[3]

2. Instalando Rails:

   Si disponemos de conexión a internet el trabajo se resumen en:

   ```
   gem install rails --include-dependencies
   ```

   Para una instalación manual debemos descargarnos cada uno de los módulos:
   Los archivos deberán ser descargados en un solo directorio y los comandos ejecutados en ese directorio, tenemos que respetar el orden de instalación, ya que algunas gemas dependen de otras.

   - gem install rake-0.7.1.gem
   - gem install activesupport-1.3.1
   - gem install activerecord-1.14.4.gem
   - gem install actionpack-1.12.5.gem
   - gem install actionmailer-1.2.5.gem
   - gem install actionwebservice-1.1.6.gem

- gem install rails-1.1.6.gem

Nota: la opción de XAMPP + AxleGrease también sirve para Linux.

6. Entornos de Desarrollo

- RADRails (multiplataforma)
- Ride – Me (Windows)
- Emacs + Rails Mode + mil y un agregados (Linux)
- Vim + snippetsEmu

7. Conclusiones

Con RoR podemos hacer aplicaciones web de manera fácil y rápida, orientadas a aplicaciones pequeñas y medianas. Este framework es óptimo para realizar proyectos exitosos con equipo de desarrolladores no muy grandes al estar orientado a la metodología ágil. Se espera que ha futuro cada vez se empiece a ir utilizando más y más debido a las ventajas que ofrece.

Referencias

[1] "Tutor de Ruby on Rails". *URL:*http://www.hackerdude.com/courses/rails/

[2] "El desarrollo del framework orientado al objeto" . *URL:* http://www.acm.org/crossroads/espanol/xrds7-4/frameworks.html

[3] "Introducción a Ruby on Rails|Asociación Nacional de Webmasters Perú" . *Página de Asociación Nacional de Webmasters Perú. URL:* http://www.anwmp.org/tutoriales/una-introduccion-a-ruby-on-rails

[4] Lenz, P. "Building Your Own Ruby on Rails Web Applications", SitePoint, 1er Edición, Marzo del 2007

[5] Geer, D. "Will software developers ride Ruby on Rails to success?". *Computer*, Vol. 39, No. 2. (2006), pp. 18-20.

[6] "Ruby on Rails" . *URL:* http://www.rubyonrails.org.es/

[7] "XAMPP - Wikipedia" . *URL:* http://es.wikipedia.org/wiki/XAMPP