Cómo Programar Para

ÍNDICE

INTRODUCCIÓN

A menudo la gente, especialmente para los ingenieros en computación, todo se centra o está relacionado con las máquinas. Se piensa: "Haciendo esto, la máquina funcionará más rápido. Haciendo esto, la máquina funcionará de manera más eficiente. Haciendo esto..." Están centrados en las máquinas, pero en realidad necesitamos centrarnos en las personas, en cómo hacen programas o cómo manejan las aplicaciones en los ordenadores.

“Nosotros somos los jefes. Ellos son los esclavos.”

Ruby sigue el "principio de la menor sorpresa", lo que significa que el lenguaje debe comportarse de tal manera que minimice la confusión de los usuarios experimentados. Matz (creador de Ruby) ha dicho que su principal objetivo era hacer un lenguaje que le divirtiera él mismo, minimizando el trabajo de programación y la posible confusión.

1. ¿QUÉ ES RAIL?

Rails es un framework de desarrollo de aplicaciones web, escrito en el lenguaje Ruby. Está diseñado para hacer más fácil la programación de aplicaciones web, haciendo suposiciones sobre lo que cada desarrollador necesita para comenzar.

Rails es un software obstinado. Se hace la suposición de que hay una "mejor" manera de hacer las cosas, y está diseñado para promover esa misma idea, y en algunos casos, para desalentar las posibles alternativas. Si usted se entera de *"The Rails Way"* es probable que descubra un tremendo incremento de la productividad. Y si persiste en llevar los viejos hábitos de otros idiomas para su desarrollo Rails, y tratando de utilizar patrones que aprendí en otros lugares, es posible que tenga una experiencia muy desagradable y menos feliz.

La filosofía de Rails incluye varios principios:

- **SECO**. "No Repeat Yourself" sugiere que escribir el mismo código una y otra vez es algo malo.
- **Convención sobre la configuración.** Significa que Rails hace suposiciones sobre lo que quiere hacer y cómo vas a hacerlo, en lugar de exigir que se especifique cada pequeña cosa a través de los archivos de configuración sin fin.
- **El descanso es el mejor patrón para las aplicaciones web**. La organización de su aplicación alrededor de los recursos y el nivel verbos HTTP es el mejor camino a seguir.

1.1. La arquitectura MVC

En el centro de Rails es el modelo, Vista, Controlador de la arquitectura, por lo general sólo son llamados MVC. Los beneficios de MVC son:

- El aislamiento de la lógica empresarial de la interfaz de usuario.
- Facilidad de mantenimiento de la sequedad de código.
- Dejando claro que los diferentes tipos de código pertenecen a un mantenimiento más fácil.

1.1.1. Modelos

Un modelo representa la información (datos) de la aplicación y las normas para manipular esos datos. En el caso de los Rails, los modelos se utilizan principalmente para la gestión de las reglas de interacción con una tabla de base de datos correspondiente.

En la mayoría de los casos, una tabla en la base de datos corresponde a un modelo en su aplicación. La mayor parte de la lógica de negocio de su aplicación se concentra en los modelos.

1.1.2. Vistas

Representan puntos de vista de la interfaz de usuario de su aplicación. En Rails, los puntos de vista son a menudo los archivos HTML con código Ruby que realizan tareas relacionadas únicamente con la presentación de los datos. Las vistas hacen el trabajo de proporcionar datos para el navegador web u otra herramienta que se utiliza para hacer peticiones desde la aplicación.

1.1.3. Controladores

Los controladores proporcionan el "pegamento" entre los modelos y los puntos de vista. En Rails, los controladores son responsables de procesar las peticiones entrantes desde el navegador web, interrogando a los modelos de datos, y que pasa los datos sobre los puntos de vista para la presentación.

1.2. Los componentes de Rails

Buques Rails como muchos de los componentes individuales.

- Action Pack
  - Action Controller
  - Action Dispatch
  - Action View
- Action Mailer
- Active Model
- Active Record
- Active Resource
- Active Support
- Railties

1.2.1. Action Pack

Paquete de Acción es una joya única que contiene Action Controller, Vista Acción y Despacho de acción. El "VC" parte de "MVC".

1.2.2. Action Controller

Acción del controlador es el componente que administra los controladores en una aplicación Rails. El marco de acción del controlador procesa las solicitudes entrantes a una aplicación Rails, extractos de los parámetros, y las envía a la acción prevista.

Los servicios prestados por la acción del controlador incluyen la administración de sesiones, lo que hace de plantilla, y la gestión de redireccionamiento.

1.2.3. Action Dispatch

Acción Despacho se encarga del enrutamiento de las peticiones web y las despacha como desee, ya sea para su aplicación o cualquier otra aplicación de rack.

1.2.4. Action View

Acción Ver gestiona los puntos de vista de su aplicación Rails. Se pueden crear ambos XML y HTML de salida por default. Acción Ver gestiona plantillas de representación, incluyendo plantillas anidadas y parcial, e incluye built-in AJAX support.

1.2.5. Action Mailer

Acción Mailer es un marco para la creación de servicios de correo electrónico. Puede utilizar la acción Mailer para recibir y procesar el correo entrante y enviar mensajes de texto sin formato simple o complejo, correos electrónicos de varias partes sobre la base de plantillas flexibles.

1.2.6. Active Model

Modelo Activo proporciona una interfaz definida entre los Action Pack gem services y Object Relationship Mapping gems como Active Record. Active Model permite que los Rails utilicen otros marcos ORM en lugar de Active Record si su aplicación necesita esto.

1.2.7. Active Record

Es la base para los modelos en una aplicación Rails. Proporciona independencia de base de datos, la funcionalidad básica de CRUD, avanzadas capacidades de búsqueda, y la capacidad de relacionar los modelos entre sí, entre otros servicios.

1.2.8. Active Resource

Proporciona un marco para la gestión de la conexión entre los objetos de negocio y servicios web RESTful. Se implementa a los recursos de un mapa basado en web los objetos locales con la semántica CRUD.

1.2.9. Active Support

Es una extensa colección de clases de utilidades y extensiones de la biblioteca estándar de Ruby que se utilizan en los Riels, tanto por el código del núcleo como de sus aplicaciones.

1.2.10. Railties

Railties es el código del núcleo Rails que construye aplicaciones Rails nuevas colas y los distintos marcos y complementos juntos en cualquier aplicación Rails.

2. ORIGENES

Ruby es un lenguaje de programación interpretado, reflexivo y orientado a objetos, creado por el programador japonés Yukihiro "Matz" Matsumoto, quien comenzó a trabajar en Ruby en 1993, y lo presentó públicamente en 1995.

Combina una sintaxis inspirada en Python y Perl con características de programación orientada a objetos similares a Smalltalk. Comparte también funcionalidad con otros lenguajes de programación como Lisp, Lua, Dylan y CLU.

Ruby es un lenguaje de programación interpretado en una sola pasada y su implementación oficial es distribuida bajo una licencia de software libre.

2.1. Historia

En el círculo de amigos de Matsumoto se le puso el nombre de "Ruby" (en español rubí) como broma aludiendo al lenguaje de programación "Perl" (perla). La última versión estable de la rama 1.8 es la 1.8.7_p248, de la rama 1.9 es la 1.9.1_p378.

La versión en 1.9 que incorpora mejoras sustanciales en el rendimiento del lenguaje, que se espera queden reflejadas en la próxima versión estable de producción del lenguaje, Ruby 1.9.0.1 Diferencias en rendimiento entre la actual implementación de Ruby (1.8.6) y otros lenguajes de programación más arraigados han llevado al desarrollo de varias máquinas virtuales para Ruby. Entre ésas se encuentra JRuby, un intento de llevar Ruby a la plataforma Java, y Rubinius, un intérprete modelado basado en las máquinas virtuales de Smalltalk.

Los principales desarrolladores han apoyado la máquina virtual proporcionada por el proyecto YARV, que se fusionó en el árbol de código fuente de Ruby el 31 de diciembre de 2006, y se dió a conocer como Ruby 1.9.

3. REQUERIMIENTOS PARA DESARROLLO DE PROGRAMAS

3.1. Interacción

La distribución oficial de Ruby incluye "irb"(Interactive Ruby Shell), un intérprete interactivo de línea de comandos que puede ser usado para probar código de manera rápida. El siguiente fragmento de código representa una muestra de una sesión usando irb:

```
$ irb
irb(main):001:0> puts "Hola mundo"
Hola mundo
=> nil
irb(main):002:0> 1+2
=> 3
```

3.2. Sintaxis

La sintaxis de Ruby es similar a la de Perl o Python. La definición de clases y métodos está definida por palabras clave. Sin embargo, en Perl, las variables no llevan prefijos. Cuando se usa, un prefijo indica el ámbito de las variables. La mayor diferencia con C y Perl es que las palabras clave son usadas para definir bloques de código sin llaves. Los saltos de línea son significativos y son interpretados como el final de una sentencia, el punto y coma tiene el mismo uso. De forma diferente que Python, la indentación no es significativa.

Una de las diferencias entre Ruby y Python y Perl es que Ruby mantiene todas sus variables de instancia privadas dentro de las clases y solo la expone a través de métodos de acceso (attr_writer, attr_reader, etc). A diferencia de los métodos "getter" y "setter" de otros lenguajes como C++ o Java, los métodos de acceso en Ruby pueden ser escritos con una sola línea de código. Como la invocación de estos métodos no requiere el uso de paréntesis, es trivial cambiar una variable de instancia en una función sin tocar una sola línea de código o refactorizar dicho código. Los descriptores de propiedades de Python son similares pero tienen una desventaja en el proceso de desarrollo.

Si uno comienza en Python usando una instancia de variable expuesta públicamente y después cambia la implementación para usar una instancia de variable privada expuesta a través de un descriptor de propiedades, el código interno de la clase necesitará ser ajustado para usar la variable privada en vez de la propiedad pública.

Ruby elimina está decisión de diseño obligando a todas las variables de instancia a ser privadas, pero también proporciona una manera sencilla de declarar métodos set y get. Esto mantiene el principio de que en Ruby no se puede acceder a los miembros internos de una clase desde fuera de esta, en lugar de esto se pasa un mensaje (se invoca un método) a la clase y recibe una respuesta.

4. CREAR UN NUEVO PROYECTO RAILS

4.1. Crear la aplicación 'blog'

Para poder ver el código completo de esta aplicación, se puede descargar de Aquí.

1. Para empezar a crear la app, abra un terminal, navegue a la carpeta donde tiene los derechos para crear archivos y escriba:

```
<span onmouseover="_tipon(this)" onmouseout="_tipoff()"><span class="google-src-text" style="direction: ltr; text-align: left">$ rails new blog</span> $ Carriles nuevo blog</span>
```

Esto creará una aplicación Rails llamado blog en un directorio llamado blog.

Puedes ver todos los interruptores que el generador de aplicaciones Rails acepta mediante la ejecución de `los carriles-h.`

2. Después de crear la aplicación de blog, cambie a la carpeta para continuar el trabajo directamente en esta solicitud:

```
<span onmouseover="_tipon(this)" onmouseout="_tipoff()"><span class="google-src-text" style="direction: ltr; text-align: left">$ cd blog</span> $ Cd blog</span>
```

En cualquier caso, Rails creará una carpeta en su directorio de trabajo llamado `blog.` Abre la carpeta y puede explorar su contenido.

4.1.1. Instalación de los Gems requeridos

Administrar las aplicaciones Rails con dependencia en gema se logra atreves de Bundler de forma predeterminada. Como no se necesita ninguna otra gema más allá de los generados en el Gemfile se puede ejecutar directamente para tenerlas listas:

```
<span onmouseover="_tipon(this)" onmouseout="_tipoff()"><span class="google-
src-text" style="direction: ltr; text-align: left">bundle install</span>
paquete de instalación</span>
```

4.2. Configuración de una base de datos

Casi todas las aplicaciones Rails van a interactuar con una base de datos. La base de datos a utilizar se especifica en un archivo de configuración config / database.yml. Si abre este archivo en una nueva aplicación Rails, puedes encontrar una configuración de base de datos predeterminada utilizando SQLite3. El archivo contiene secciones para los tres entornos diferentes en los que Rails se puede ejecutar de forma predeterminada:

- ✓ El entorno de desarrollo se utiliza en el equipo a medida que interactúan de forma manual con la aplicación.
- ✓ El entorno de prueba se utiliza para ejecutar pruebas automatizadas.
- ✓ El entorno de producción se utiliza cuando se implementa la aplicación para usarlo en cualquier parte del mundo.

4.2.1. Configuración de una base de datos MySQL

Si usted elige utilizar MySQL en vez de la base de datos sqlite3 antes mencionada, su fichero de configuración / database.yml se verá un poco diferente. Aquí está la sección de desarrollo:

```
<span onmouseover="_tipon(this)" onmouseout="_tipoff()"><span class="google-src-
text" style="direction: ltr; text-align: left">development:</span> el
desarrollo:</span>
  <span onmouseover="_tipon(this)" onmouseout="_tipoff()"><span class="google-src-
text" style="direction: ltr; text-align: left">adapter: mysql2</span> adaptador:
mysql2</span>
  <span onmouseover="_tipon(this)" onmouseout="_tipoff()"><span class="google-src-
text" style="direction: ltr; text-align: left">encoding: utf8</span> codificación:
UTF-8</span>
  <span onmouseover="_tipon(this)" onmouseout="_tipoff()"><span class="google-src-
text" style="direction: ltr; text-align: left">database: blog_development</span>
base de datos: blog_development</span>
  <span onmouseover="_tipon(this)" onmouseout="_tipoff()"><span class="google-src-
text" style="direction: ltr; text-align: left">pool: 5</span> Piscina: 5</span>
  <span onmouseover="_tipon(this)" onmouseout="_tipoff()"><span class="google-src-
text" style="direction: ltr; text-align: left">username: root</span> nombre de
usuario: root</span>
  <span onmouseover="_tipon(this)" onmouseout="_tipoff()"><span class="google-src-
text" style="direction: ltr; text-align: left">password:</span> contraseña:</span>
  <span onmouseover="_tipon(this)" onmouseout="_tipoff()"><span class="google-src-
text" style="direction: ltr; text-align: left">socket: /tmp/mysql.sock</span>
socket: / tmp / mysql.sock</span>
```

Si la instalación de su equipo de desarrollo de MySQL incluye un usuario *root* con una contraseña vacía, esta configuración debe funcionar para usted. De lo contrario, cambiar el nombre de usuario y contraseña en la sección de desarrollo, según convenga.

5. INSTALACIÓN Y EJECUCIÓN DE APLICATIVOS

5.1. Instalación de Rails

Si está trabajando en Windows, debe tener en cuenta que la gran mayoría del desarrollo de Rails se realiza en entornos Unix. Mientras Ruby y Rails se instalan fácilmente utilizando, por ejemplo el instalador Ruby, el ecosistema de apoyo a menudo se asume que son capaces de construir con sede en rubygems C y trabajar en una ventana de comandos. La mejor forma de programar en Rails es creando una máquina virtual de Linux y así el uso en el desarrollo de Rails será más efectivo, en lugar de usar Windows.

En la mayoría de los casos, la forma más fácil de instalar Rails es tomar ventaja de RubyGems:

```
<span onmouseover="_tipon(this)" onmouseout="_tipoff()"><span class="google-
src-text" style="direction: ltr; text-align: left">Usually run this as the
root user:</span> Por lo general, ejecutar como usuario root:</span>

<span onmouseover="_tipon(this)" onmouseout="_tipoff()"><span class="google-
src-text" style="direction: ltr; text-align: left"># gem install
rails</span> # Carriles joya instalar</span>
```

CONCLUSIÓN

En conclusión se pudo observar los aspectos más sobresalientes acerca de cómo programar para Ruby on Rails. Sin olvidar mencionar que este documento solo ha abarcado los aspectos importantes en el inicio de la creación de un programa en Ruby, es decir, solo se ha dado la base para crear el programa 'blog'.

REFERENCIAS

•**Getting Started with Rails**.

•**Ruby on Rails Tutorial**.

•**Ruby on Rails guides**.

•**API documentation**.