

Ruby, Rails, RubyMine

Material preparado por: R. Casallas

rcasalla@uniandes.edu.co

Departamento de Sistemas y Computación

Universidad de los Andes, Colombia

El lenguaje de programación Ruby

Referencias:

<http://www.rubyist.net/~slagell>
<http://phrogz.net/ProgrammingRuby/>

...

Lo básico ...

- ▶ “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.
 - ▶ Su implementación oficial es distribuida bajo una licencia de software libre.” **Wikipedia**

Orientado-objetos

- ▶ Todos los tipos de datos son un objeto (clases y tipos que primitivos (enteros, cadenas de caracteres, booleanos, y "nil")
- ▶ Soporta tipado dinámico y polimorfismo
- ▶ Toda función es un método.
- ▶ Las variables siempre son referencias a objetos
- ▶ Soporta herencia con enlace dinámico
- ▶ No soporta herencia múltiple
- ▶ Tiene el concepto de módulos que son colecciones de métodos mixins. Estos pueden ser incluidos en las clases

37 razones ...

(<http://rubyhacker.com/ruby37.html>)

- | | |
|--|--|
| 1. It's object-oriented. | 20. It uses punctuation and capitalization creatively. |
| 2. It's a pure OOP language | 21. Reserved words aren't. |
| 3. It's a dynamic language. | 22. It allows iterators. |
| 4. It's an interpreted language. | 23. It has safety and security features. |
| 5. It understands regular expressions. | 24. It has no pointers. |
| 6. It's multi-platform. | 25. It pays attention to detail |
| 7. It's derivative. | 26. It has a flexible syntax. |
| 8. It's innovative. | 27. It has a rich set of libraries. |
| 9. It's a Very High-Level Language (VHLL | 28. It has a debugger. |
| 10. It has a smart garbage collector. | 29. It can be used interactively. |
| 11. It's a scripting language. | 30. It is concise. |
| 12. It's versatile. | 31. It is expression-oriented. |
| 13. It's thread-capable. | 32. It is laced with syntax sugar |
| 14. It's open-source. | 33. It has operator overloading. |
| 15. It's intuitive. | 34. It has infinite-precision integer arithmetic. |
| 16. It has an exception mechanism. | 35. It has an exponentiation operator. |
| 17. It has an advanced Array class. | 36. It has powerful string handling. |
| 18. It's extensible. | 37. It has few exceptions to its rules. |
| 19. It encourages literate programming. | |

Ejemplo: Calcular factorial

La declaración

```
# Program to find the  
factorial of a number  
# Save this as fact.rb
```

```
def fact(n)  
  if n == 0  
    1  
  else  
    n * fact(n-1)  
  end  
end
```

```
puts fact(ARGV[0].to_i)
```

La invocación

```
% ruby fact.rb 40  
81591528324789773434561126959611  
5894272000000000
```

Método definido para ninguna clase o todas las clases (Object).

Ni a las variables ni a los parámetros se les define un tipo (es dinámico)

Un método siempre retorna la valor de la última expresión que evaluó (por eso no hay necesidad de escribir return)

“puts” escribe en un archivo (fact.rb)

Iteradores

► Métodos para las colecciones

```
collection.each do |variable|  
  code  
end
```

```
ary = [1,2,3,4,5] ary.each do |i|  
  puts i  
  
end
```

Produce el siguiente resultado:

```
1  
2  
3  
4  
5
```

Clases, Objetos y Variables

La declaración

```
class Song
  def initialize(name, artist,
                duration)
    @name = name
    @artist = artist
    @duration = duration
  end
end
```

Definición de una clase y su constructor.

Una variable de instancia es un nombre precedido de @

La invocación

```
aSong = Song.new("Bicylops", "Fleck", 260)

aSong.inspect →
  "#<Song:0x401b4924 @duration=260,
  @artist=\"Fleck\", @name=\"Bicylops\">
```

new para invocar el constructor

aSong es una instancia de la clase Song

inspect es un método de la clase Object

Atributos (read-only)

```
class Song
  def name
    @name
  end
  def artist
    @artist
  end
  def duration
    @duration
  end
end
```

```
aSong = Song.new("Bicylops", "Fleck", 260)
aSong.artist → "Fleck"
aSong.name → "Bicylops"
aSong.duration → 260
```

Atributos (read-write)

```
class Song
  def duration=(newDuration)
    @duration = newDuration
  end
end

aSong = Song.new("Bicylops", "Fleck", 260)
aSong.duration → 260
aSong.duration = 257 # set attribute with updated value
aSong.duration → 257
```

```
class Song
  attr_writer :duration
end

aSong = Song.new("Bicylops", "Fleck", 260)
aSong.duration = 257
```

La herencia

```
class KaraokeSong < Song
  def initialize(name, artist,
                duration, lyrics)
    super(name, artist, duration)
    @lyrics = lyrics
  end
end
```

KaraokeSong es una subclase de la clase Song

El constructor de KaraokeSong utiliza el constructor de la superclase Song

La herencia

```
class Song
  ...
  def to_s
    "Song: #{@name}--
      #{@artist} (#{@duration})"
  end
end
```

Los métodos de la superclase se pueden redefinir en la subclase

```
class KaraokeSong < Song
  ...
  def to_s
    super + " [#{@lyrics}]"
  end
end
```

Rails

Referencias:
<http://ruby.railstutorial.org>

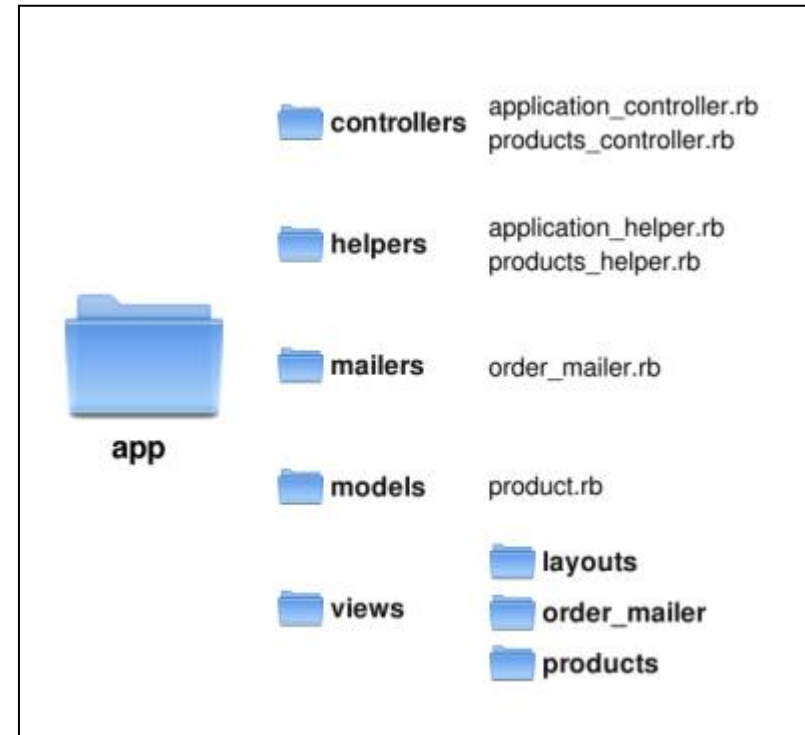
Lo básico

- ▶ “Ruby on Rails, también conocido como RoR o Rails es un framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby, siguiendo el paradigma de la arquitectura Modelo Vista Controlador (MVC).
- ▶ Trata de combinar la simplicidad con la posibilidad de desarrollar aplicaciones del mundo real escribiendo menos código que con otros frameworks y con un mínimo de configuración.
- ▶ El lenguaje de programación Ruby permite la metaprogramación, de la cual Rails hace uso, lo que resulta en una sintaxis que muchos de sus usuarios encuentran muy legible.
- ▶ Rails se distribuye a través de RubyGems, que es el formato oficial de paquete y canal de distribución de bibliotecas y aplicaciones Ruby.” **Wikipedia**

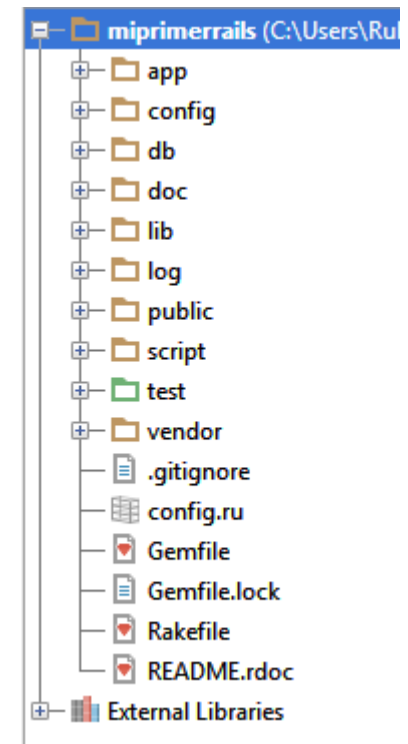
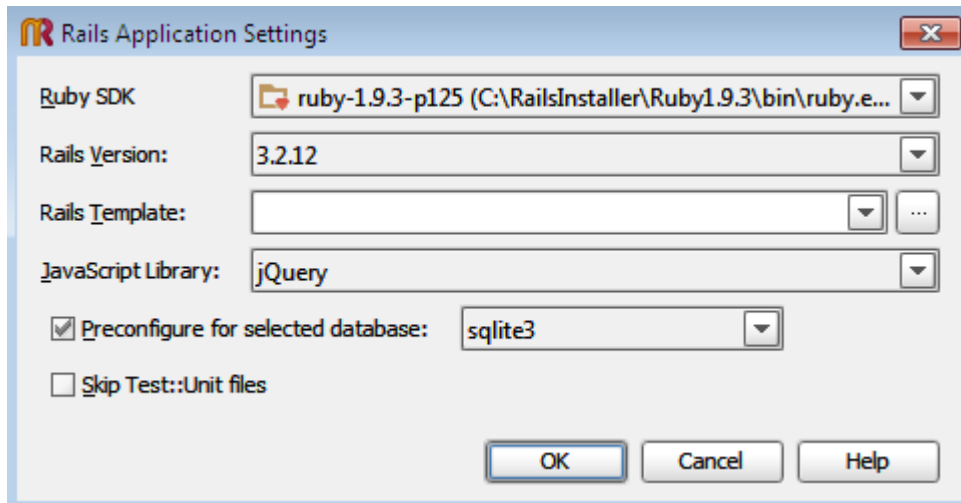
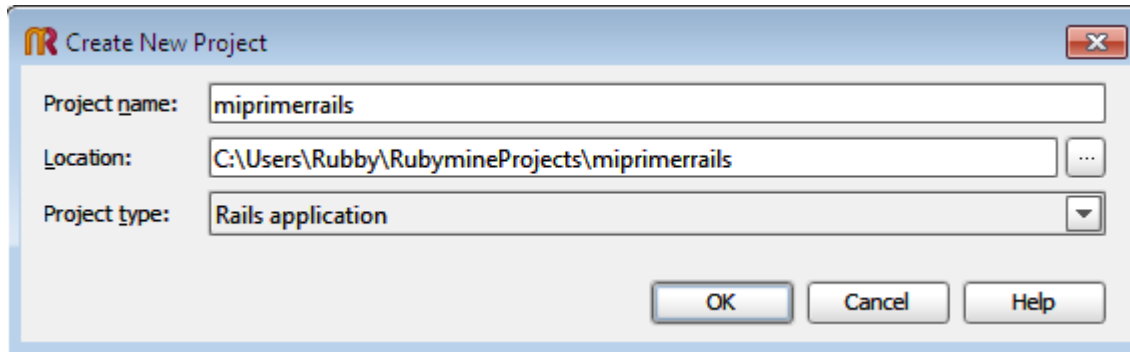
Principios de diseño de Rails

- ▶ Active Record Framework
- ▶ MVC: Models – Views – Controllers
- ▶ Don't repeat yourself, DRY:
 - ▶ Las definiciones se deben hacer una sola vez.
 - ▶ No escribir el mismo código en distintos lugares (Utilizar helpers y librerías)
- ▶ Priman las convenciones sobre las configuraciones:
 - ▶ Evitar muchos archivos de configuración
 - ▶ Por ejemplo: si la clase se llama User, la tabla en la base de datos se llamará users.

Estructura de un proyecto en RoR



Un proyecto en RoR



Active Record Framework

- ▶ Maneja el mapping entre el modelo y la persistencia en la base de datos
- ▶ A cada clase le corresponde una tabla
- ▶ El framework genera el código que se ocupa de realizar las operaciones CRUD
- ▶ Las instancias de la clase corresponde a filas en la tabla de la BD
- ▶ Cuando una instancia es actualizada en memoria, está se actualiza en la base de datos

MVC in Rails

► Modelo:

- Esta capa contiene el código que opera sobre los datos de la aplicación. Por ejemplo, las operaciones CRUD son escritas en esta capa.

► Vistas:

- Es la capa de presentación. Define:
- cómo se mostrarán las páginas al usuario final.
- cómo la aplicación presenta los datos
- cómo el usuario ingresar datos o peticiones a la aplicación

► Controlador:

- controla el flujo del programa. Recibe comandos del usuario, contacta al modelo e interactúa con las vistas para presentar los resultados.

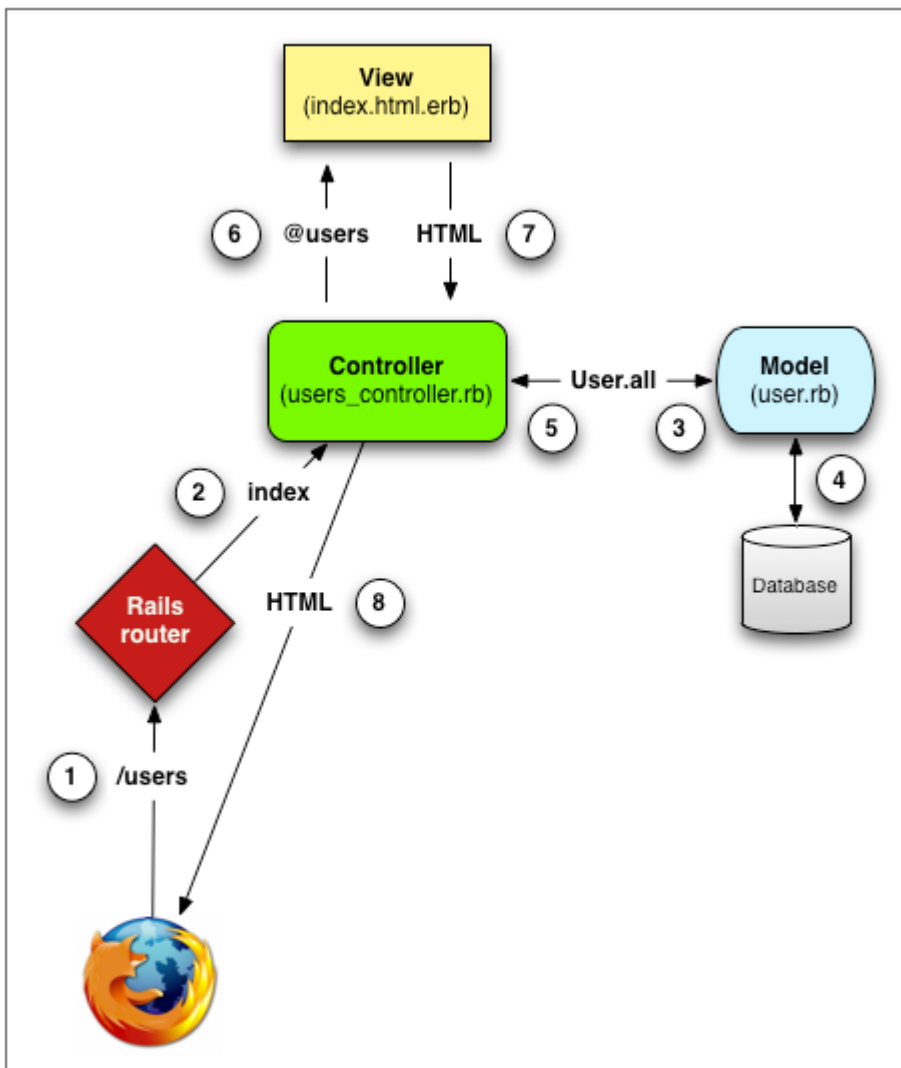


Figure 2.11: A detailed diagram of MVC in Rails. [\(full size\)](#)

1. El usuario realiza desde un browser un request a la aplicación
2. El enrutador de rails que se encuentra en el servidor web, enruta al controlador correspondiente
3. Dependiendo de la operación, el controlador invoca al modelo
4. EL modelo probablemente acceda la base de datos
5. La respuesta es enviada al controlador desde el modelo
6. El Controlador envía los datos para ser desplegados en la vista
7. Regresa el html al controlador
8. Se despliega en el browser

MVC en la estructura del proyecto

myaplicacion/
app/

controllers/

models/

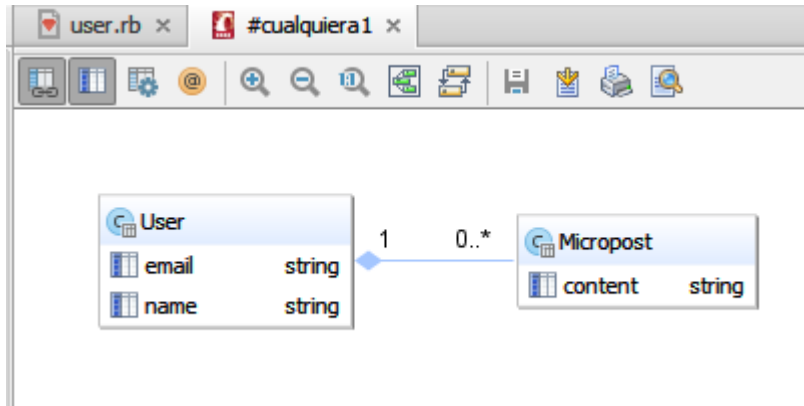
views/

Contiene archivos en código ruby (.rb)

Contiene templates para ser transformados a html (.erb)

MVC component	Rails Sublibrary	Propósito
Model	ActiveRecord	Provee una interface y el binding entre las tablas de la base de datos relacional y el código ruby que maneja los registros en la base de datos. Los métodos para hacer esto son generados por Rails
View	ActionView	Provee servicios para desplegar información dentro de las páginas html en forma de embedded ruby code erb
Controller	ActionController	Provee facilidades invocar servicios del modelo y/o las vistas y para desplegar la información que devuelven las vistas

Ejemplo: Model



app/models/user.rb

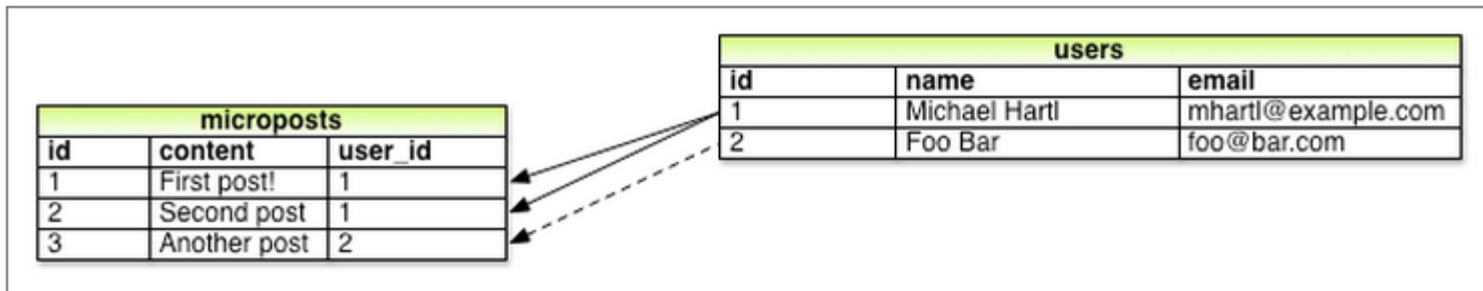
```
class User < ActiveRecord::Base
  attr_accessible :email, :name
  has_many :microposts
end
```

app/models/micropost.rb

```
class Micropost < ActiveRecord::Base
  attr_accessible :content, :user_id

  belongs_to :user

  validates :content, :length => { :maximum => 140 }
end
```



<http://ruby.railstutorial.org/chapters/a-demo-app?version=3.2#top>

Ejemplo: Controller

Listing 2.3. The Users controller in schematic form.
`app/controllers/users_controller.rb`

```
class UsersController < ApplicationController

  def index
    .
    .
    .
  end

  def show
    .
    .
    .
  end

  def new
    .
    .
    .
  end

end
```

Listing 2.8. The Microposts controller in schematic form.
`app/controllers/microposts_controller.rb`

```
class MicropostsController < ApplicationController

  def index
    .
    .
    .
  end

  def show
    .
    .
    .
  end

  def new
    .
    .
    .
  end

end
```

`config/routes.rb`

```
DemoApp::Application.routes.draw do
  resources :microposts
  resources :users
  .
  .
  .
end
```

Ejemplo:

View **Listing 2.6.** The view for the user index.

`app/views/users/index.html.erb` →

```
<h1>Listing users</h1>
```

```
<table>
```

```
  <tr>
```

```
    <th>Name</th>
```

```
    <th>Email</th>
```

```
    <th></th>
```

```
    <th></th>
```

```
    <th></th>
```

```
  </tr>
```

```
<% @users.each do |user| %>
```

```
  <tr>
```

```
    <td><%= user.name %></td>
```

```
    <td><%= user.email %></td>
```

```
    <td><%= link_to 'Show', user %></td>
```

```
    <td><%= link_to 'Edit', edit_user_path(user) %></td>
```

```
    <td><%= link_to 'Destroy', user, method: :delete,  
                                data: { confirm: 'Are you sure?' } %></td>
```

```
  </tr>
```

```
<% end %>
```

```
</table>
```

```
<br />
```

```
<%= link_to 'New User', new_user_path %>
```

erb = embedded ruby:

`<% ruby code %>` Ejecuta el código

`<%= ruby code %>` Imprime el resultado de la ejecución

Link_to método definido en:
Module ActionView::Helpers::UrlHelper

Ejemplo: Controller

```
class UsersController < ApplicationController
  # GET /users
  # GET /users.json
  def index
    @users = User.all

    respond_to do |format|
      format.html # index.html.erb
      format.json {render json: @users}
    end
  ...
end
```

1. Desde el browser se invocó:
.../users o
.../users.json

2. El route de rails localiza el
UserController y la acción
index

3. @users = User.all invoca el
modelo y pide toda la lista de
usuarios que hay en la base de
datos

4. Prepara la respuesta .
Dependiendo de como se
llamó, esta puede ser:
- Resolver la página que se
encuentra en index.html.erb
- Devolver en formato json
los datos de los usuarios



Listing users

Name	Email	
Michael Hartl	michael@example.org	Show Edit Destroy
Foo Bar	foo@bar.com	Show Edit Destroy

[New User](#)

Qué es un Gem?

- ▶ Es una librería empaquetada en un archivo zip o tar.
- ▶ Para empaquetarlo hay que respetar una configuración
- ▶ Una gema tiene tres componentes:
 - ▶ La especificación de la gema
 - ▶ La documentación
 - ▶ El código

Gemfile

```
source 'https://rubygems.org'
```

```
gem 'rails', '3.2.12'
```

Para rails

```
gem 'sqlite3'
```

Para la base de datos

Project Settings [cualquiera1]

- + Code Style
 - Coverage
- + Deployment
 - File Colors
 - Inspections
- + JavaScript
- + Project Structure
 - Ruby SDK and Gems**
- + Schemas and DTDs
 - Scopes
 - Spelling
 - SQL Dialects
- + Tasks
- + Version Control

IDE Settings

- Appearance
- Console Folding
- + Debugger
 - Diagrams
- + Editor
 - Extensions
 - External Diff Tools
 - External Tools
 - File Templates
 - File Types
 - General
 - HTTP Proxy
 - Images
 - Intentions
 - Keymap
 - Live Templates
 - Menus and Toolbars
 - Notifications
 - Passwords

Ruby SDK and Gems

Ruby SDK:

ruby-1.9.3-p125 (C:\RailsInstaller\Ruby1.9.3\...

Add SDK...

Language level:

version 1.9










Remove SDK

RVM gemset:

[n/a]

☐ Ignore global gempath

Installed Gems

Name	Versions
 actionmailer	3.2.12, 3.2.1
 actionpack	3.2.12, 3.2.1
 activemodel	3.2.12, 3.2.1
 activerecord	3.2.12, 3.2.1
 activerecord-sqlserver-adapter	3.2.10, 3.2.1
 activeresource	3.2.12, 3.2.1
 activesupport	3.2.12, 3.2.1
 arel	3.0.2
 builder	3.2.0, 3.1.4, 3.0.4, 3.0.0

Description

Install Gems...

Update Gems...

OK

Cancel

Apply

Help

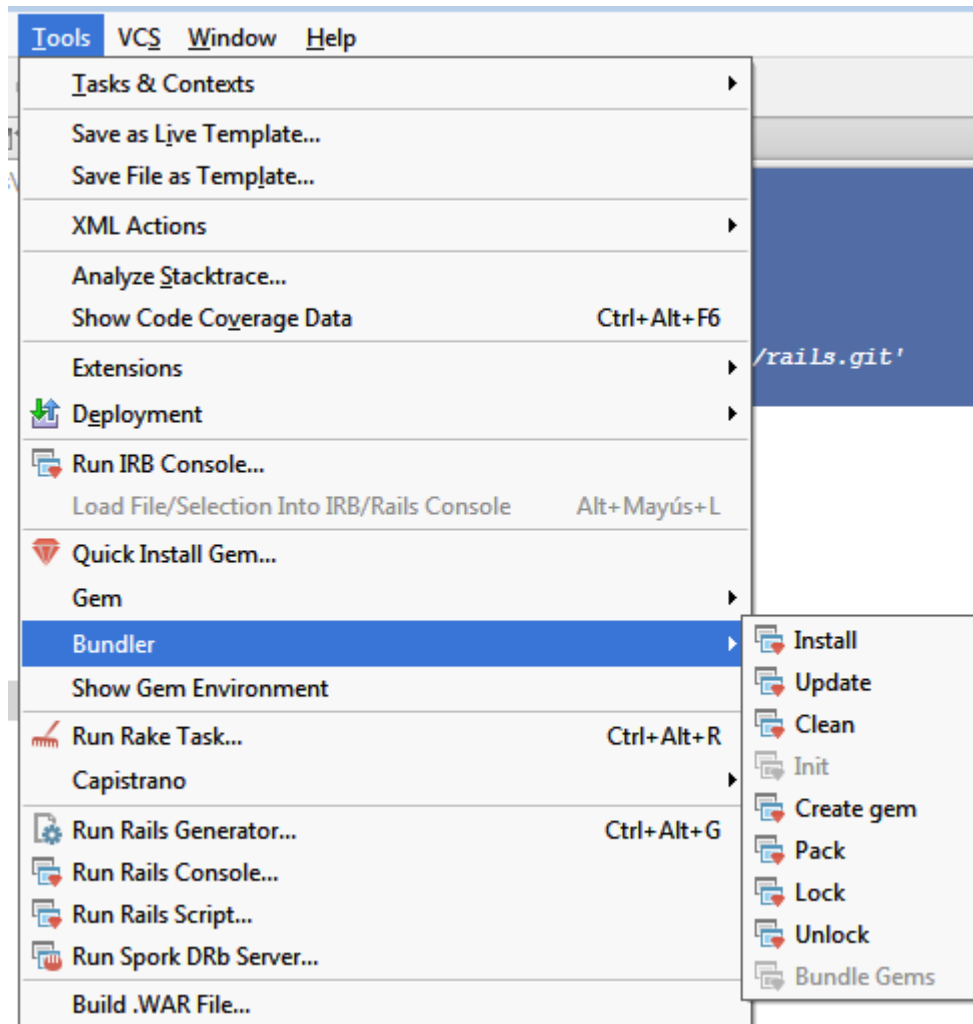
The best way to manage your
application's dependencies

Bundler



What is Bundler?

Bundler maintains a consistent environment for ruby applications. It tracks an application's code and the rubygems it needs to run, so that an application will always have the exact gems (and versions) that it needs to run.



Andamios o scaffolds

- ▶ Es una utilidad que permite crear de manera automática algunos elementos de la aplicación
- ▶ Recibe de entrada el nombre del modelo (clase y atributos) y crea los componentes básicos del MVC


```
$ rails generate scaffold User name:string email:string
  invoke  active_record
  create  db/migrate/20111123225336_create_users.rb
  create  app/models/user.rb
  invoke  test_unit
  create  test/unit/user_test.rb
  create  test/fixtures/users.yml
  route  resources :users
  invoke  scaffold_controller
  create  app/controllers/users_controller.rb
  invoke  erb
  create  app/views/users
  create  app/views/users/index.html.erb
  create  app/views/users/edit.html.erb
  create  app/views/users/show.html.erb
  create  app/views/users/new.html.erb
  create  app/views/users/_form.html.erb
  invoke  test_unit
  create  test/functional/users_controller_test.rb
  invoke  helper
  create  app/helpers/users_helper.rb
  invoke  test_unit
  create  test/unit/helpers/users_helper_test.rb
  invoke  assets
  invoke  coffee
  create  app/assets/javascripts/users.js.coffee
  invoke  scss
  create  app/assets/stylesheets/users.css.scss
  invoke  scss
  create  app/assets/stylesheets/scaffolds.css.scss
```

