An Introduction to Ruby on Rails

Outline

- What is Ruby?
- What is Rails?
- Rails overview.
- Sample RoR application and also with Spree gem.

What is Ruby?

- Ruby is a pure object oriented programming language. It was created on February 24, 1993 by Yukihiro Matsumoto of Japan. Ruby is a general-purpose, interpreted programming language.

What is Rails?

- Rails is a web application development framework written in the Ruby language. It is designed to make programming web applications easier by making assumptions about what every developer needs to get started.
- Open Source (MIT license)
- Based on a existing application (Basecamp)
- Provides common needs:

  - Routing, sessions

  - Database storage

  - Business logic

  - Generate HTML/XML/CSS/Ajax
- It allows you to write less code while accomplishing more than many other languages and frameworks.

Rails Advantages

- Convention over configuration
- Don’t Repeat Yourself
- Object Relational Mapping
- Model View Controller
- Reuse of code

Convention over Configuration

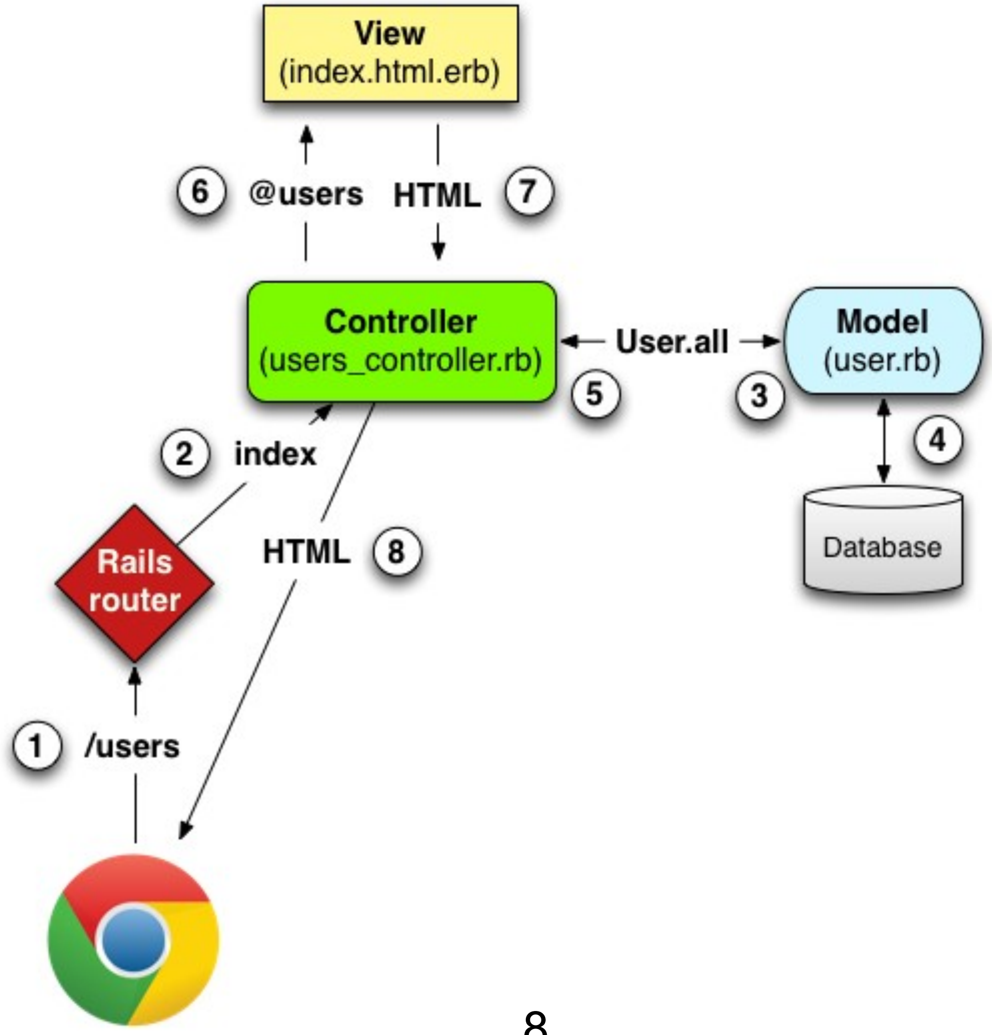
- Table and foreign key naming
  - Tables are multiples (users, orders, …)
  - Foreign key naming: user_id
- Default locations
  - MVC, Test, Languages, Plugins
- Naming
  - Class names: CamelCase
  - Files: lowercase_underscored.rb

Don’t repeat yourself

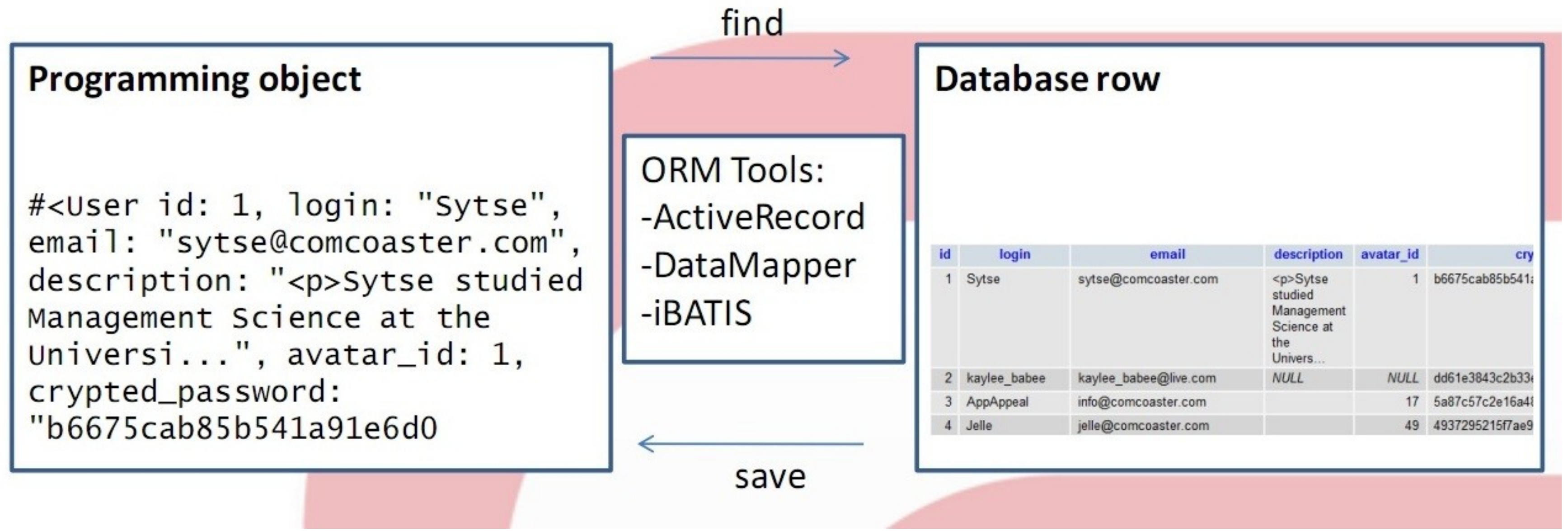
- Everything is defined in a single, unambiguous place
- Easier to find code
  - Only need to look once
  - Can stop looking when found
  - Well defined places for most items
- Much easier to maintain code
  - Faster to change
  - Less inconsistency bugs

MVC

- Model
  - Object relationships (users, orders)
- Controller
  - Business logic (perform a payment)
- View
  - Visual representation (generate HTML/XML)



Object Relational Mapping



```
#<User id: 1, login: "Sytse",
email: "sytse@comcoaster.com",
description: "<p>Sytse studied
Management Science at the
Universi...", avatar_id: 1,
crypted_password:
"b6675cab85b541a91e6d0
```

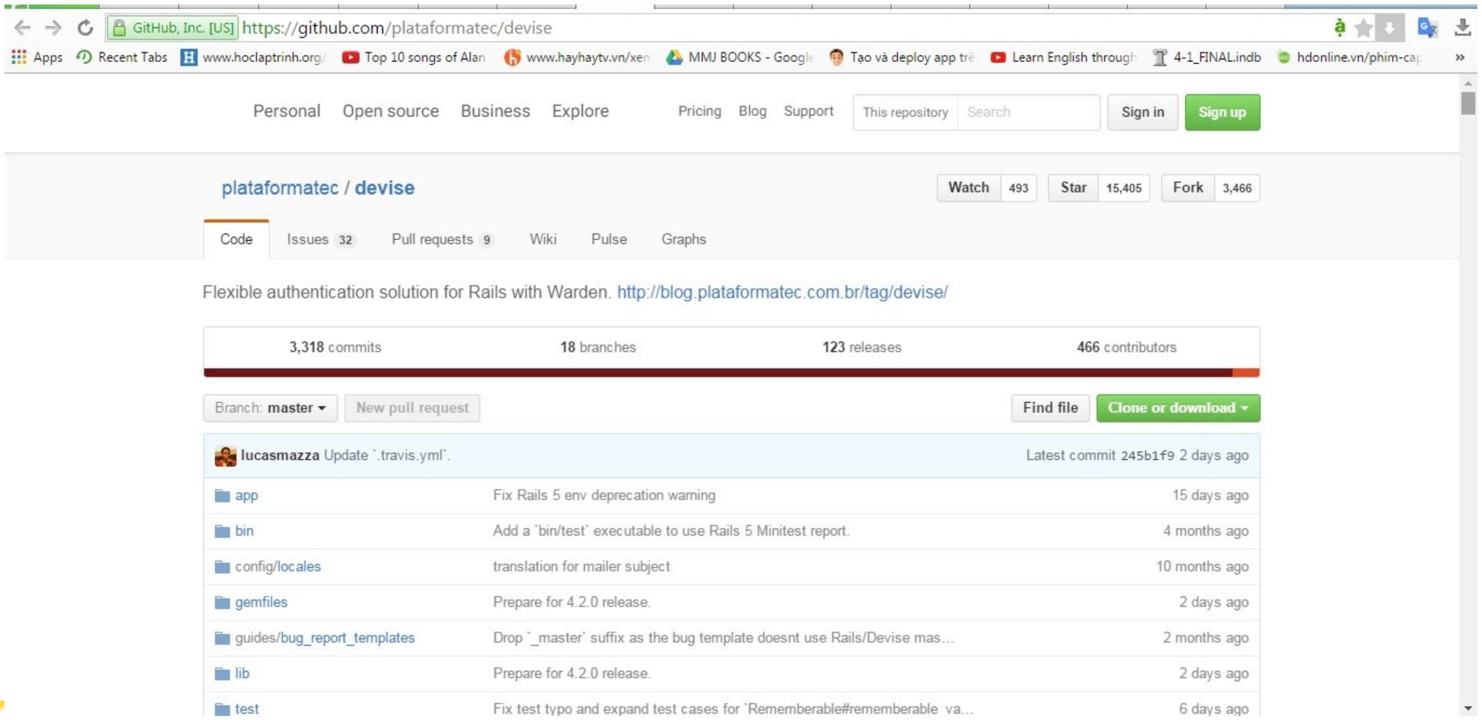
id	login	email	description	avatar_id	cry
1	Sytse	sytse@comcoaster.com	<p>Sytse studied Management Science at the Univers...	1	b6675cab85b541
2	kaylee_babee	kaylee_babee@live.com	NULL	NULL	dd61e3843c2b33
3	AppAppeal	info@comcoaster.com		17	5a87c57c2e16a4
4	Jelle	jelle@comcoaster.com		49	4937295215f7ae9

- Easily stored and retrieved from a database without writing SQL statements directly
- Use with less overall database access code

```
# Examples of finding records
User.find(:all)
User.find(23).articles
User.find_by_first_name('Johnny')
User.order.find(:last).lines_items.count
```

Re-use of code

- Gems and plugins, more then 1300
  - For authentication, pagination, testing, …
- Git allows easy forking and merging



plataformatec / devise

Flexible authentication solution for Rails with Warden. http://blog.plataformatec.com.br/tag/devise/

3,318 commits · 18 branches · 123 releases · 466 contributors

lucasmazza Update `.travis.yml`. Latest commit 245b1f9 2 days ago

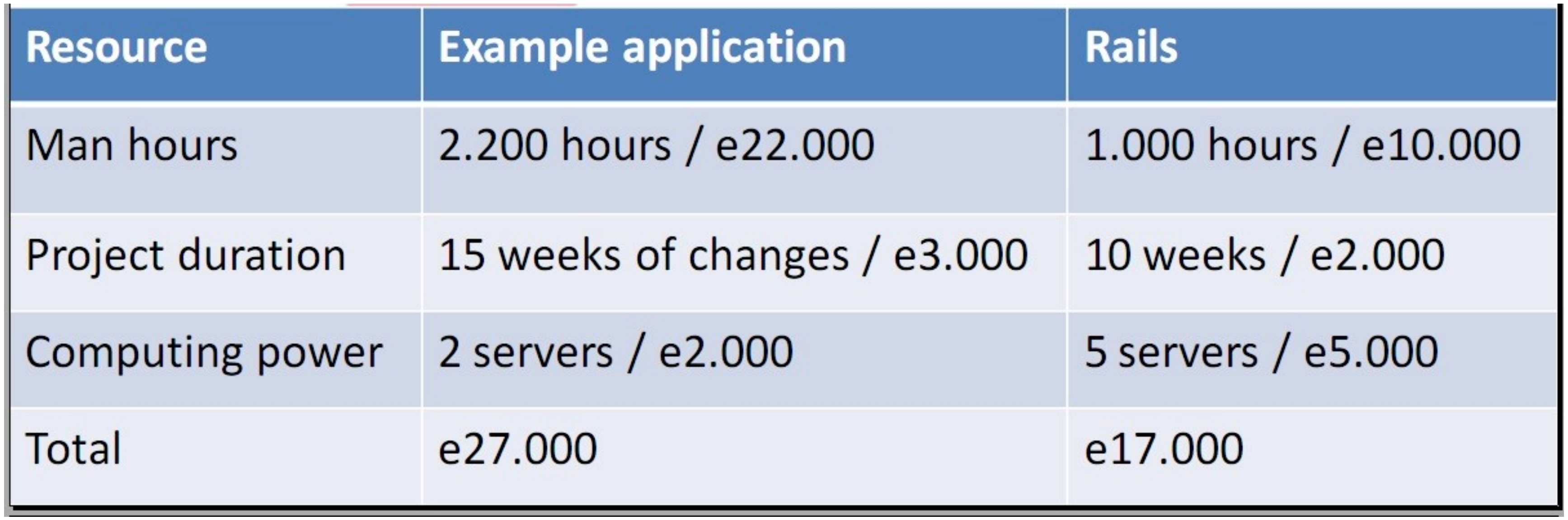
app	Fix Rails 5 env deprecation warning	15 days ago
bin	Add a `bin/test` executable to use Rails 5 Minitest report.	4 months ago
config/locales	translation for mailer subject	10 months ago
gemfiles	Prepare for 4.2.0 release.	2 days ago
guides/bug_report_templates	Drop `_master` suffix as the bug template doesnt use Rails/Devise mas…	2 months ago
lib	Prepare for 4.2.0 release.	2 days ago
test	Fix test typo and expand test cases for `Rememberable#rememberable_va…	6 days ago

Rails Disadvantages

- Rails is inefficient
- Rails is hard to desploy

Rails is inefficient

- Rails uses a lot of memory, up to 150MB per instance
- Requires more application servers
- Where are your development constrains?



Resource	Example application	Rails
Man hours	2.200 hours / e22.000	1.000 hours / e10.000
Project duration	15 weeks of changes / e3.000	10 weeks / e2.000
Computing power	2 servers / e2.000	5 servers / e5.000
Total	e27.000	e17.000

Rails is hard to deploy

- Harder than PHP, better with Passenger
- Lots of moving parts
  - Rails, apps, gems, plugins
  - Application server, webserver
- Deployment via scripts
  - Via the Capistrano tool
  - Easy to break something
- Deployment to multiple servers

Quick Rails Demo:

- $ rails new RoRApp

  $ cd RoRApp

(Use an Aptana studio IDE)

- We have 3 environments(in config/database.yml)

    1.Development

    2.Test

    3.Production

Configuring the Database:

- If we want to configure the database as mysql2(default database is sqllite), open the file config/database.yml and modify the database name and options.

```
development:
  adapter: mysql2
  encoding: utf8
  database: db/development/dev
  username: root
  Password: '123456'
```

- And also we need to add myspl2 gem in Gemfile as

  `gem 'mysql2'`

Now we need to bundle update which installs the mysql2 gem for our application.

```
$ bundle update
```

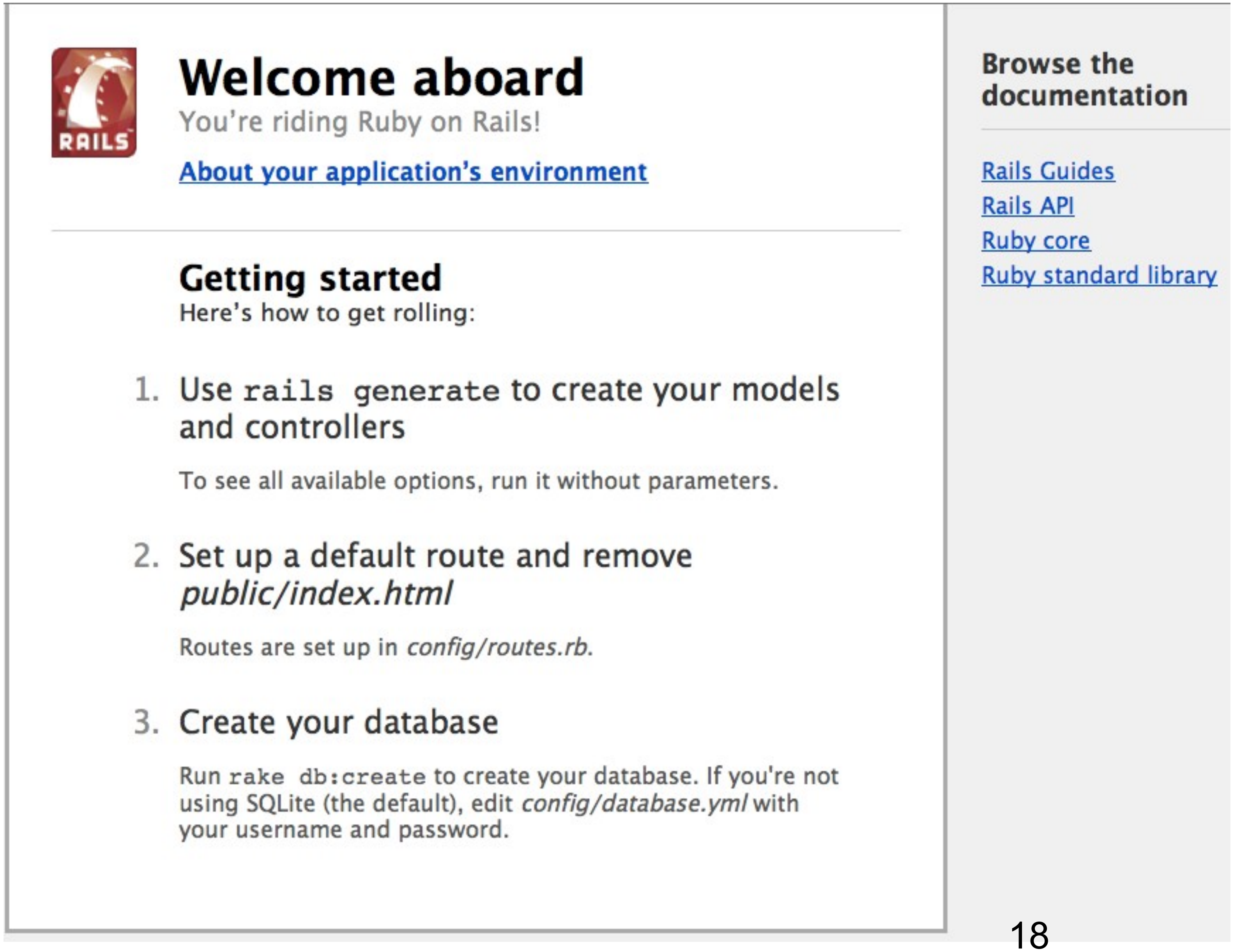
Create a Database :

- Now we have our database is configured, it’s time to have Rails create an empty database. We can do this by running a rake command:

```
$ rake db:create
```

1. rails s (or) (default port is 3000)
2. rails server (or)
3. rails s -p 4000

- Verify whether the rails app is working properly or not by browsing http://localhost:3000
- Here the default page is rendering from public/index.html



Welcome aboard

You're riding Ruby on Rails!

About your application's environment

Getting started

Here's how to get rolling:

1. Use `rails generate` to create your models and controllers

   To see all available options, run it without parameters.

2. Set up a default route and remove *public/index.html*

   Routes are set up in *config/routes.rb*.

3. Create your database

   Run `rake db:create` to create your database. If you're not using SQLite (the default), edit *config/database.yml* with your username and password.

Browse the documentation

- Rails Guides
- Rails API
- Ruby core
- Ruby standard library

- If we want to display a user defined text or anything in our home page, we need to create a controller and a view

```
$ rails generate controller home index
```

Rails will create several files, including
app/views/home/index.html.erb and
app/controllers/home_controller.rb

- To make this index file as the home page, 1st we need to delete the default public/index.html page

```
$ rm public/index.html
```

- Now, we have to tell Rails where your actual home page is located. For that open the file config/routes.rb and edit as

```
root :to => "home#index"
```

- Check whether our home page is rendering proper page or not, for that we need to start the rails serve as

```
$ rails s
```

Scaffolding

Rails scaffolding is a quick way to generate some of the major pieces of an application. If you want to create the models, views, and controllers for a new resource in a single operation, scaffolding is the tool for the job.

Example:

Creating a Resource for sessionApp:

We can start by generating a scaffold for the Post resource: this will represent a single blog posting.

```
$  rails generate scaffold Post name:string title:string content:text
```

File	Purpose
db/migrate/20120606184725_create_posts.rb	Migration to create the posts table in your database (your name will include a different timestamp)
app/models/post.rb	The Post model
config/routes.rb	Edited to include routing information for posts
app/controllers/posts_controller.rb	The Posts controller
app/views/posts/index.html.erb	A view to display an index of all posts
app/views/posts/edit.html.erb	A view to edit an existing post
app/views/posts/show.html.erb	A view to display a single post
app/views/posts/new.html.erb	A view to create a new post

Running a Migration:

- Rails generate scaffold command create is a database migration. Migrations are Ruby classes that are designed to make it simple to create and modify database tables. Rails uses rake commands to run migrations, and it’s possible to undo a migration ($ rake db:migrate rollback) after it’s been applied to your database.

- If we look in the db/migrate/20120606184725_create_posts.rb

```
class CreatePosts < ActiveRecord::Migration
  def change
    create_table :posts do |t|
      t.string :name
      t.string :title
      t.text :content
      t.timestamps
  end
end
```

- At this point, we can use a rake command to run the migration:

```
$ rake db:migrate
```

Rails will execute this migration command and tell you it created the Posts table.

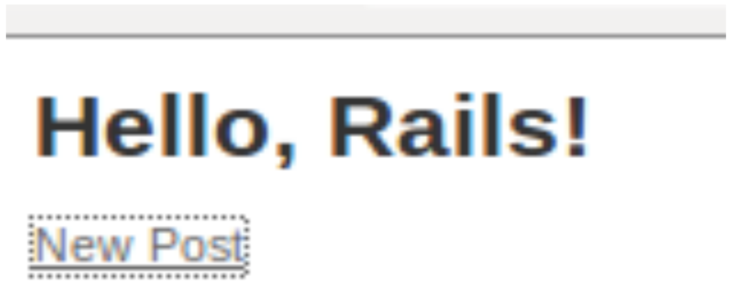
```
==  CreatePosts: migrating
    ===================================================
    =
-- create_table(:posts)
   -> 0.0019s
==  CreatePosts: migrated (0.0020s)
    ===========================================
```

Adding a Link:

- We can add a link to the home page. Open app/views/home/index.html.erb and modify it as follows:

```
<h1>Hello, Rails!</h1>
<%= link_to "New Post", posts_path %>
```

When we run the server it displays the home page as



Hello, Rails!

New Post

The Model:

- The model file, app/models/post.rb is

```
class Post < ActiveRecord::Base
  attr_accessible :content, :name, :title
end
```

- Active Record supplies a great deal of functionality to our Rails models for free, including basic database CRUD (Create, Read, Update, Destroy) operations, data validation

Adding Some Validation:

- Rails includes methods to help you validate the data that you send to models. Open the app/models/post.rb file and edit it:

```
class Post < ActiveRecord::Base
      attr_accessible :content, :name, :title
      validates :name,  :presence => true
            validates :title, :presence => true,
                        :length => {:minimum => 5}
end
```

Listing All Posts

- How the application is showing us the list of Posts.
- Open the file app/controllers/posts_controller.rb and look at the index action:

```
def index
    @posts = Post.all
    respond_to do |format|
    format.html  # index.html.erb
    format.json  { render :json => @posts }
    end
end
```

- The HTML format(format.html) looks for a view in app/views/posts/ with a name that corresponds to the action name(index). Rails makes all of the instance variables from the action available to the view. Here’s app/views/posts/index.html.erb:

```
<h1>Listing posts</h1>
<table>
  <tr>  <th>Name</th>
            <th>Title</th>
            <th>Content</th>
            <th></th>
            <th></th>
        <th></th>
  </tr>
<% @posts.each do |post| %>
  <tr>
            <td><%= post.name %></td>
        <td><%= post.title %></td>
        <td><%= post.content %></td>
        <td><%= link_to 'Show', post %></td>
        <td><%= link_to 'Edit', edit_post_path(post) %></td>
        <td><%= link_to 'Destroy', post, :confirm => 'Are you sure?',
                                    :method => :delete %></td>
  </tr>
<% end %>
</table>
<br />
<%= link_to 'New post', new_post_path %>
```