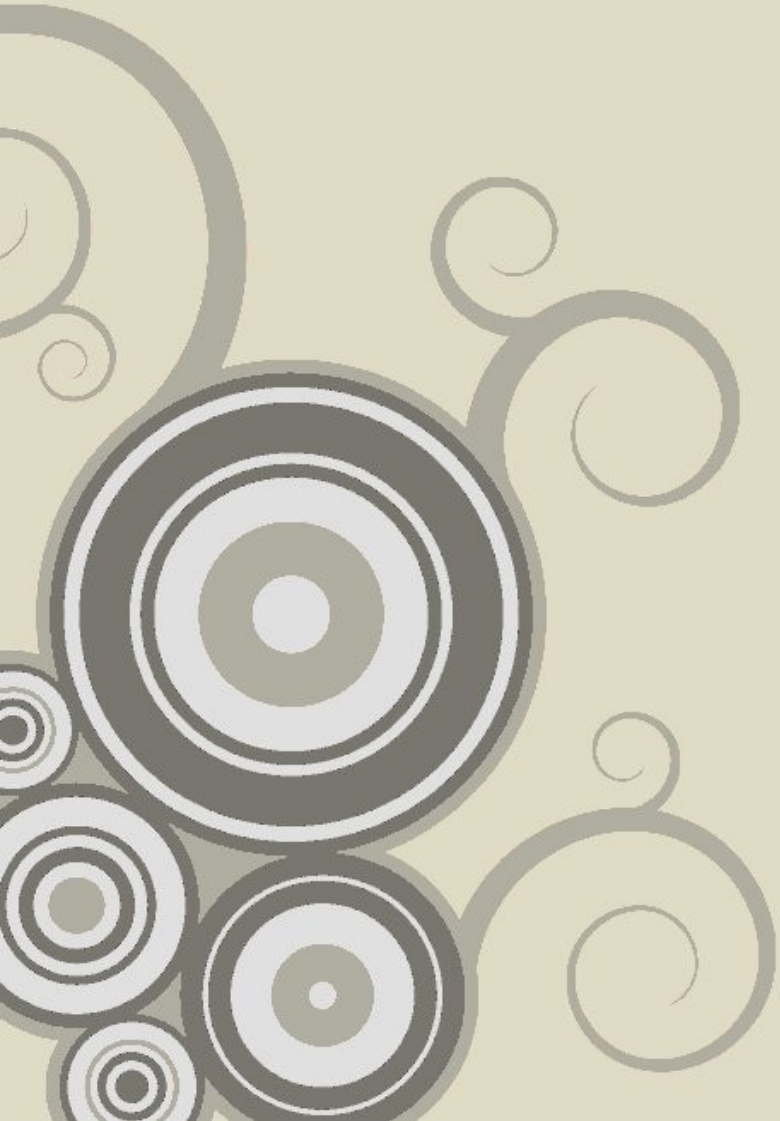
Laboratorio de Computación IV



Clase 10

Andrés Fortier

Antes de comenzar

- ¿Error de javascript en Rails?

Comandos del día: tree

- tree == Muestra los contenidos de un directorio en forma recursiva

```
$ cd ejemplorails
$ tree
.
├── app
│   ├── assets
│   │   ├── images
│   │   ├── javascripts
│   │   │   └── application.js
│   │   └── stylesheets
│   │       └── application.css
...
└── vendor
    └── assets
        ├── javascripts
        └── stylesheets
```

Comandos del día: tree

- Usando un path

```
$ tree app/controllers/
app/controllers/
├── application_controller.rb
├── articles_controller.rb
├── concerns
└── example_controller.rb

1 directory, 3 files
```

- -f → todo el path

```
$ tree -f app/controllers/
$ tree -f | grep controller
```

Comandos del día: tree

- -i → sin indentación

```
$ tree -fi | grep controller
./app/controllers
./app/controllers/application_controller.rb
./app/controllers/articles_controller.rb
./app/controllers/concerns
./app/controllers/example_controller.rb
```

- -P → Filtra usando un patrón

```
$ tree -P "*.rb"
```

- --prune → No mostrar directorio vacíos

```
$ tree -P "*.rb" --prune
```

Repaso

- Llegamos a tener el índice de artículos en http://localhost:3000/

/app/views/articles/index.html.erb

```
<h1> Indice </h1>
```

- Ahora necesitamos tener un modelo persistente

RoR – Modelos y Persistencia

- Rails utiliza
  - Una BD relacional para persistir datos.
  - Un mecanismo para “mapear” objetos a la BD.
    - Un poco sobre ORMs.
- *ActiveRecord:* Implementación del patrón con el mismo nombre.

  An object that wraps a row in a database table or view, encapsulates the database access, and adds domain logic on that data.

ActiveRecord

Cuenta
titular saldo
all() where(filter) ... save() delete() … depositar(unMonto) ...

titular	saldo
“Juan Perez”	$1.500,00
“Jose Julio”	$300.000,00
“Anastacio Ponce”	$12,50

```
c = Cuenta.new(“Pepe”)
c.save
c.depositar(100)
c.save
Cuenta.all
Cuenta.where({titular: “Pepe”})
```

ActiveRecord

- *Pros*
  - Muy rápido para lograr prototipos.
  - Coceptualmente simple.
  - Agrega validaciones en forma sencilla.
  - Muy conveniente para modelos basados en CRUDs y con poca lógica de negocios.

ActiveRecord

- *Cons*
  - De los ORMs en general: *Impedance mismatch*.
  - Viola SRP (*Single responsibility principle*).
  - Modelo de dominio atado a la BD
    - Puede ser poco eficiente por accesos a la BD.
      - Ej: loops y relaciones.
    - Complicado/lento para testear.

RoR - *Migrations*

- BD
  - Esquema.
  - Datos.
- Problema para manejar una BD:
  - En desarrollo alguien agrega una tabla.
  - Replicar ese cambio en otros ambientes de desarrollo.
  - Replicar ese cambio en producción.
  - Sin perder datos!

RoR - *Migrations*

- *Migration*: un cambio incremental a la BD.
- Generalmente esquema, pero a veces también datos.
- Ejemplo

```
class CreateProducts < ActiveRecord::Migration
  def change
    create_table :products do |t|
      t.string :name
      t.text :description
    end
  end
end
```

RoR - *Migrations*

- Ventajas
  - No hay que escribir SQL
    - Pero no vivimos en un mundo ideal, a veces hay que hacerlo
  - RoR no aplica dos veces la misma *migration*.
  - Versionamiento de la BD en código.
  - Muchas veces son reversibles automáticamente.

Blog - Modelo

- Crear un modelo con ActiveRecord usando un generador

```
$ bin/rails generate model Article title:string text:text
      invoke  active_record
      create    db/migrate/20150418202101_create_articles.rb
      create    app/models/article.rb
```

```
$ cat app/models/article.rb
class Article < ActiveRecord::Base
end
```

Blog - Modelo

```
$ cat db/migrate/20150418202101_create_articles.rb
class CreateArticles < ActiveRecord::Migration
  def change
    create_table :articles do |t|
      t.string :title
      t.text :text

      t.timestamps
    end
  end
end
```

```
$ bin/rake db:migrate
== 20150418202101 CreateArticles: migrating ==================
-- create_table(:articles)
   -> 0.0041s
== 20150418202101 CreateArticles: migrated (0.0042s) ========
```

Blog - Modelo

```
$ bin/rake db:migrate
$
```

Blog - Modelo

```
$ cat db/schema.rb
# encoding: UTF-8
# This file is auto-generated from the current state of the
database. Instead
# of editing this file, please use the migrations feature of
Active Record to
# incrementally modify your database, and then regenerate
this schema definition.
...
```

Blog - Modelo

```
...
# Note that this schema.rb definition is the authoritative
source for your
# database schema. If you need to create the application
database on another
# system, you should be using db:schema:load, not running all
the migrations
# from scratch. The latter is a flawed and unsustainable
approach (the more migrations
# you'll amass, the slower it'll run and the greater
likelihood for issues).
#
# It's strongly recommended that you check this file into
your version control system.
...
```

Blog - Modelo

```
...
ActiveRecord::Schema.define(version: 20150418202101) do

  create_table "articles", force: true do |t|
    t.string   "title"
    t.text     "text"
    t.datetime "created_at"
    t.datetime "updated_at"
  end

end
```

Blog - Modelo

```
$ bin/rails dbconsole
SQLite version 3.8.2 2013-12-06 14:53:30
Enter ".help" for instructions
Enter SQL statements terminated with a ";"

sqlite> .tables
articles            schema_migrations

sqlite> .schema articles
CREATE TABLE "articles" (
"id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
"title" varchar(255),
"text" text,
"created_at" datetime,
"updated_at" datetime);

sqlite> select * from articles;
```

Blog - Modelo

```
sqlite> .schema schema_migrations
CREATE TABLE "schema_migrations"
("version" varchar(255) NOT NULL);
CREATE UNIQUE INDEX "unique_schema_migrations"
ON "schema_migrations" ("version");

sqlite> select * from schema_migrations;
20150418202101
```

Blog - Modelo

- En otra consola

```
$ bin/rails console
Loading development environment (Rails 4.1.8)
2.0.0-p598 :001 > post = Article.new(title: 'First post!',
text: 'Hi there')
 => #<Article id: nil, title: "First post!", text: "Hi
there", created_at: nil, updated_at: nil>
2.0.0-p598 :002 > post.save!
   (0.3ms)  begin transaction
  SQL (1.4ms)  INSERT INTO "articles" ("created_at", "text",
"title", "updated_at") VALUES (?, ?, ?, ?)
[["created_at", "2015-04-18 20:42:38.820519"],
 ["text", "Hi there"],
 ["title", "First post!"],
 ["updated_at", "2015-04-18 20:42:38.820519"]]
   (203.6ms)  commit transaction
 => true
```

Blog - Modelo

- En la consola de la BD

```
sqlite> select * from articles;
1|First post!|Hi there|2015-04-18 20:42:38.820519|2015-04-18
20:42:38.820519
```

Blog - Modelo

- En la consola de rails

```
2.0.0-p598 :004 > Article.new(title: 'Second post', text:
'Post body').save!
   (0.1ms)  begin transaction
  SQL (1.1ms)  INSERT INTO "articles" ("created_at", "text",
"title", "updated_at") VALUES (?, ?, ?, ?)  [["created_at",
"2015-04-18 20:45:46.491115"], ["text", "Post body"],
["title", "Second post"], ["updated_at", "2015-04-18
20:45:46.491115"]]
   (173.6ms)  commit transaction
 => true
```

Blog - Modelo

- En la consola de rails

```
2.0.0-p598 :005 > Article.all
  Article Load (0.4ms)  SELECT "articles".* FROM "articles"
 =>
#<ActiveRecord::Relation [#<Article id: 1, title: "First
post!", text: "Hi there", created_at: "2015-04-18 20:42:38",
updated_at: "2015-04-18 20:42:38">,
#<Article id: 2, title: "Second post", text: "Post body",
created_at: "2015-04-18 20:45:46", updated_at: "2015-04-18
20:45:46">]>
```

Blog - Artículos

- Ya tenemos lo necesario para mostrar algo en el índice

/app/controllers/articles_controller.rb

```
class ArticlesController < ApplicationController

  def index
    @articles = Article.all
  end

end
```

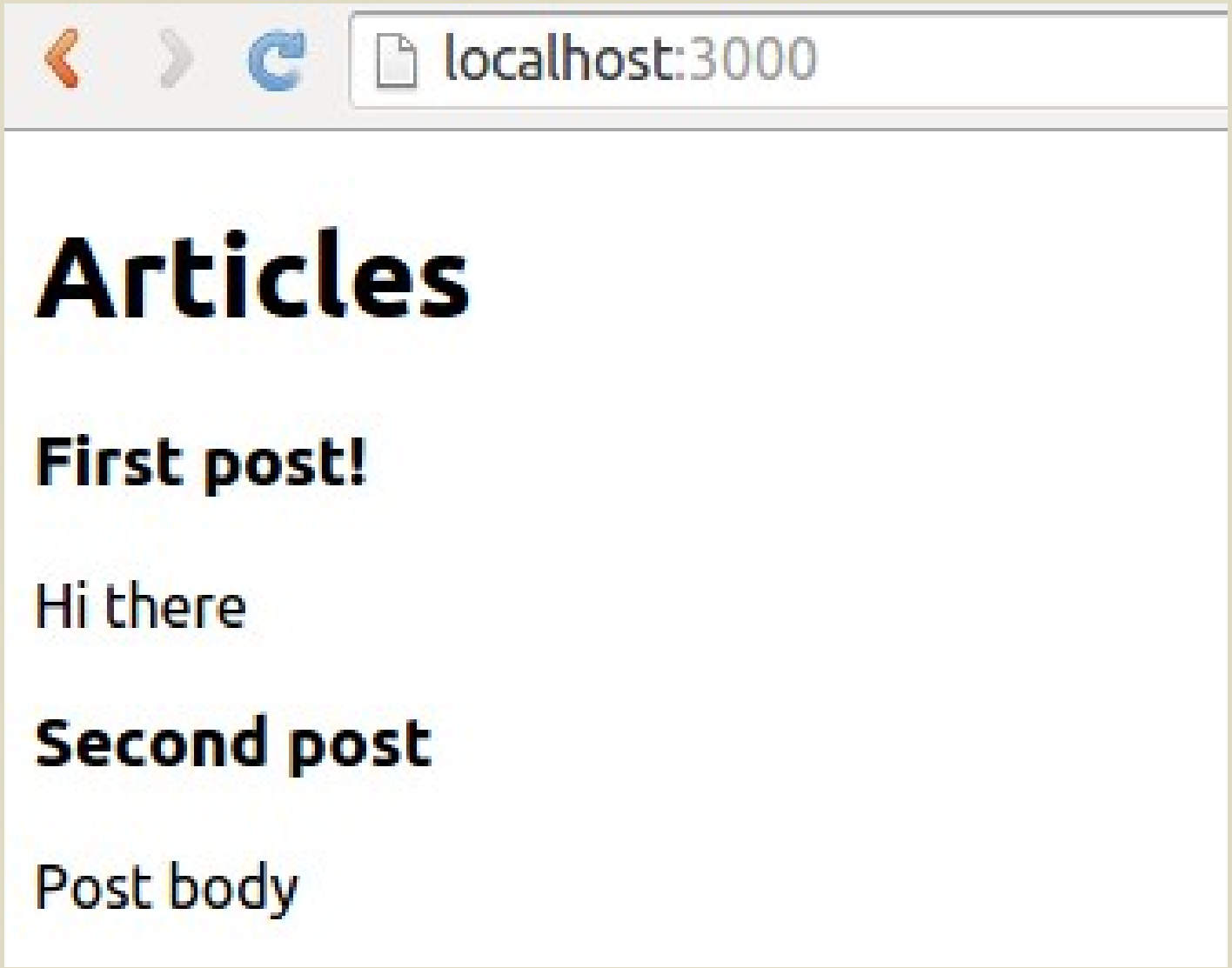
Blog - Artículos

/app/views/articles/index.html.erb

```
<h1>Articles</h1>

<% @articles.each do |article| %>
  <h3><%= article.title %></h3>
  <p> <%= article.text %> </p>
<% end %>
```

Blog - Artículos



Blog - Artículos

- Veamos ahora como linkear y ver un artículo.
  - Necesitamos una nueva ruta con un identificador.

config/routes.rb

```
Rails.application.routes.draw do
  root 'articles#index'

  get '/articles'     => 'articles#index'
  get '/articles/:id' => 'articles#show'

  get 'hello' => 'example#hello'
end
```

Blog - Artículos

- En el controller

/app/controllers/articles_controller.rb

```
class ArticlesController < ApplicationController

  ...

  def show
    @id = params[:id]
  end

end
```

Blog – Artículos

- En la vista

/app/views/articles/show.html.erb

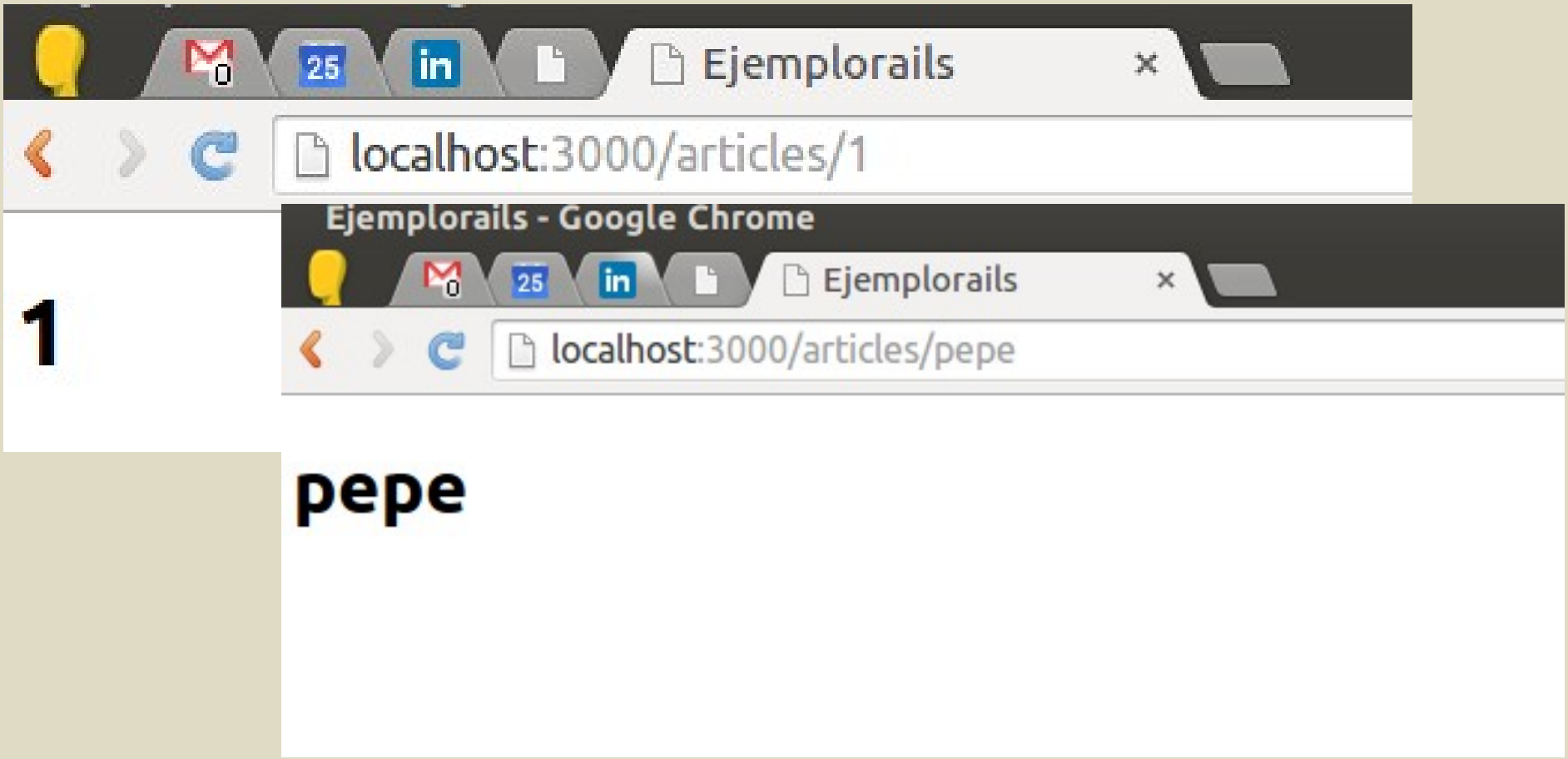
```
<h1><%= @id %></h1>
```

Blog – Artículos

- En la vista

/app/views/articles/show.erb

```
<h1><%= @id %></h1>
```



Blog - Artículos

- Volvamos al controller

/app/controllers/articles_controller.rb

```
class ArticlesController < ApplicationController

  ...

  def show
    @article = Article.find(params[:id])
  end

end
```

Blog – Artículos

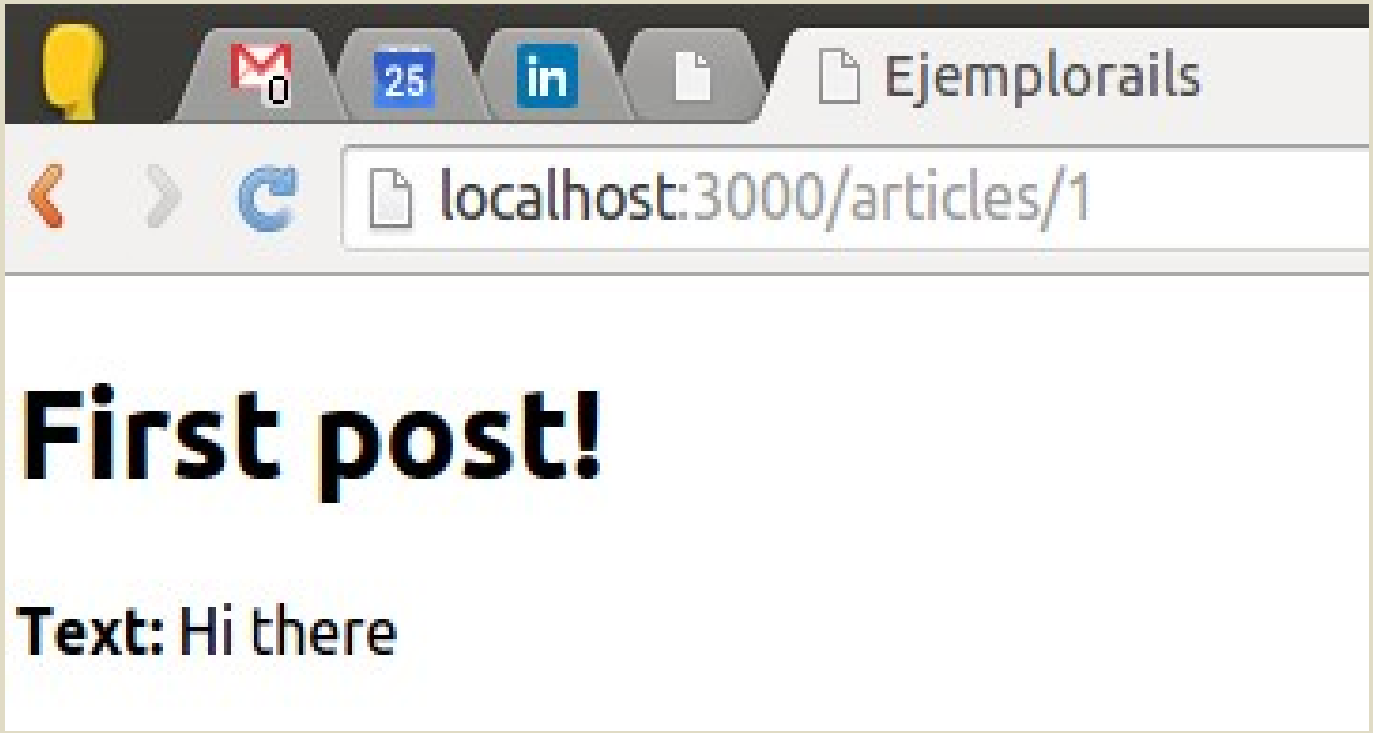
- Volvamos a la vista

/app/views/articles/show.html.erb

```
<h1>
  <%= @article.title %>
</h1>

<p>
  <strong>Text:</strong>
  <%= @article.text %>
</p>
```

Blog - Artículos



Blog - Artículos

Blog - Artículos

- Linkeemos desde el index

/app/views/articles/index.html.erb

```
<h1>Articles</h1>

<% @articles.each do |article| %>
  <h3>
    <a href="articles/<%= article.id %>">
      <%= article.title %>
    </a>
  </h3>
<% end %>
```

Blog - Artículos

- Rails provee helpers para las vistas
  - Pero previamente tenemos que darle un nombre a la ruta. Modifiquemos la definición de rutas:

config/routes.rb

```
Rails.application.routes.draw do

  root 'articles#index'

  get '/articles',
        to: 'articles#index',
        as: 'articles'
  get '/articles/:id',
        to: 'articles#show',
        as: 'article'
end
```

Blog - Artículos

- Veamos las rutas

```
$ bin/rake routes
  Prefix Verb URI Pattern                  Controller#Action
    root GET  /                            articles#index
articles GET  /articles(.:format)          articles#index
 article GET  /articles/:id(.:format)      articles#show
```

Blog – Artículos

- Ahora podemos modificar nuestra vista

/app/views/articles/index.html.erb

```
<h1>Articles</h1>

<% @articles.each do |article| %>
  <h3>
    <%= link_to article.title, article_path(article) %>
  </h3>
<% end %>
```

Blog - Artículos

- Y en la vista individual

/app/views/articles/show.html.erb

```
<h1>
  <%= @article.title %>
</h1>

<p>
  <strong>Text:</strong>
  <%= @article.text %>
</p>

<%= link_to 'Back to index', articles_path %>
```

Blog - Artículos

- Veamos como crear un artículo

config/routes.rb

```
Rails.application.routes.draw do

  get  '/articles',
         to: 'articles#index',
         as: 'articles'
  post '/articles',
         to: 'articles#create'
  get  '/articles/new',
         to: 'articles#new',
         as: 'new_article'
  get  '/articles/:id',
         to: 'articles#show',
         as: 'article'

end
```

Blog - Artículos

- Veamos las rutas

```
$ bin/rake routes
      Prefix Verb URI Pattern                  Controller#Action
        root GET  /                            articles#index
    articles GET  /articles(.:format)          articles#index
             POST /articles(.:format)          articles#create
new_article GET  /articles/new(.:format)       articles#new
     article GET  /articles/:id(.:format)      articles#show
```

Blog - Artículos

- Comencemos por el índice

/app/views/articles/index.html.erb

```
<h1>Articles</h1>

<% @articles.each do |article| %>
  <h3>
    <%= link_to article.title, article_path(article) %>
  </h3>
<% end %>

<%= link_to 'New article', new_article_path %>
```

Blog - Artículos

- En el controller

/app/controllers/articles_controller.rb

```
class ArticlesController < ApplicationController

  ...

  def new
  end

end
```

Blog – Artículos

- Y la nueva vista

/app/views/articles/new.html.erb

```
<h1>New Article</h1>

<%= form_for :article, url: articles_path do |form| %>
  <p>
    <%= form.label :title %><br>
    <%= form.text_field :title %>
  </p>
  <p>
    <%= form.label :text %><br>
    <%= form.text_area :text %>
  </p>
  <p>
    <%= form.submit %>
  </p>
<% end %>
```

Blog – Artículos

- Nos falta definir el método en el controller para crear un nuevo artículo

Blog - Artículos

/app/controllers/articles_controller.rb

```
class ArticlesController < ApplicationController

    ...

    def create
      @article = Article.new(article_params)
      @article.save
      redirect_to @article
    end

  private

    def article_params
      params.require(:article).permit(:title, :text)
    end

end
```

Pasos siguientes

- Investigar el uso de `resources :articles` en el archivo `routes.rb` en lugar de especificar las rutas en forma individual.
  - ¿Cómo podemos restringir las rutas que crea?
- Implementar
  - Borrar post
  - Editar post
- http://guides.rubyonrails.org/v4.1.8/getting_started.html

Links

- http://edgeguides.rubyonrails.org/active_record_migrations.html
- http://www.martinfowler.com/eaaCatalog/activeRecord.html
- http://guides.rubyonrails.org/active_record_basics.html
- http://guides.rubyonrails.org/command_line.html