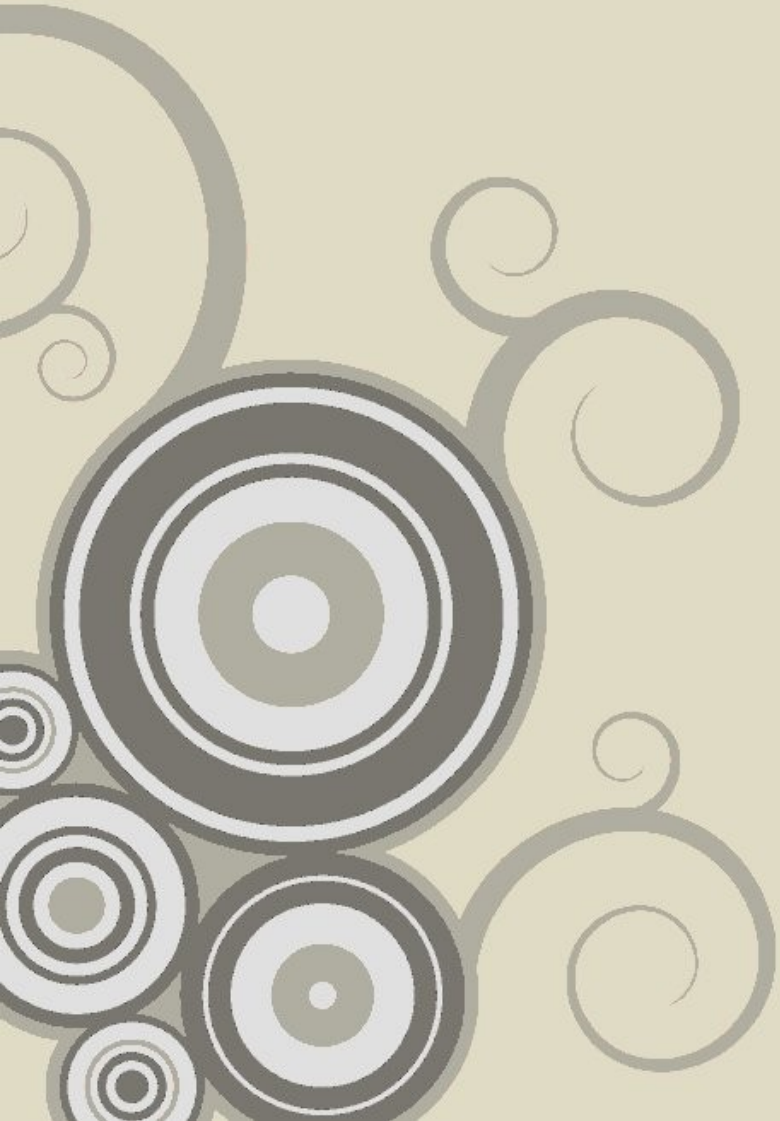
Laboratorio de Computación IV



Clase 19

Andrés Fortier

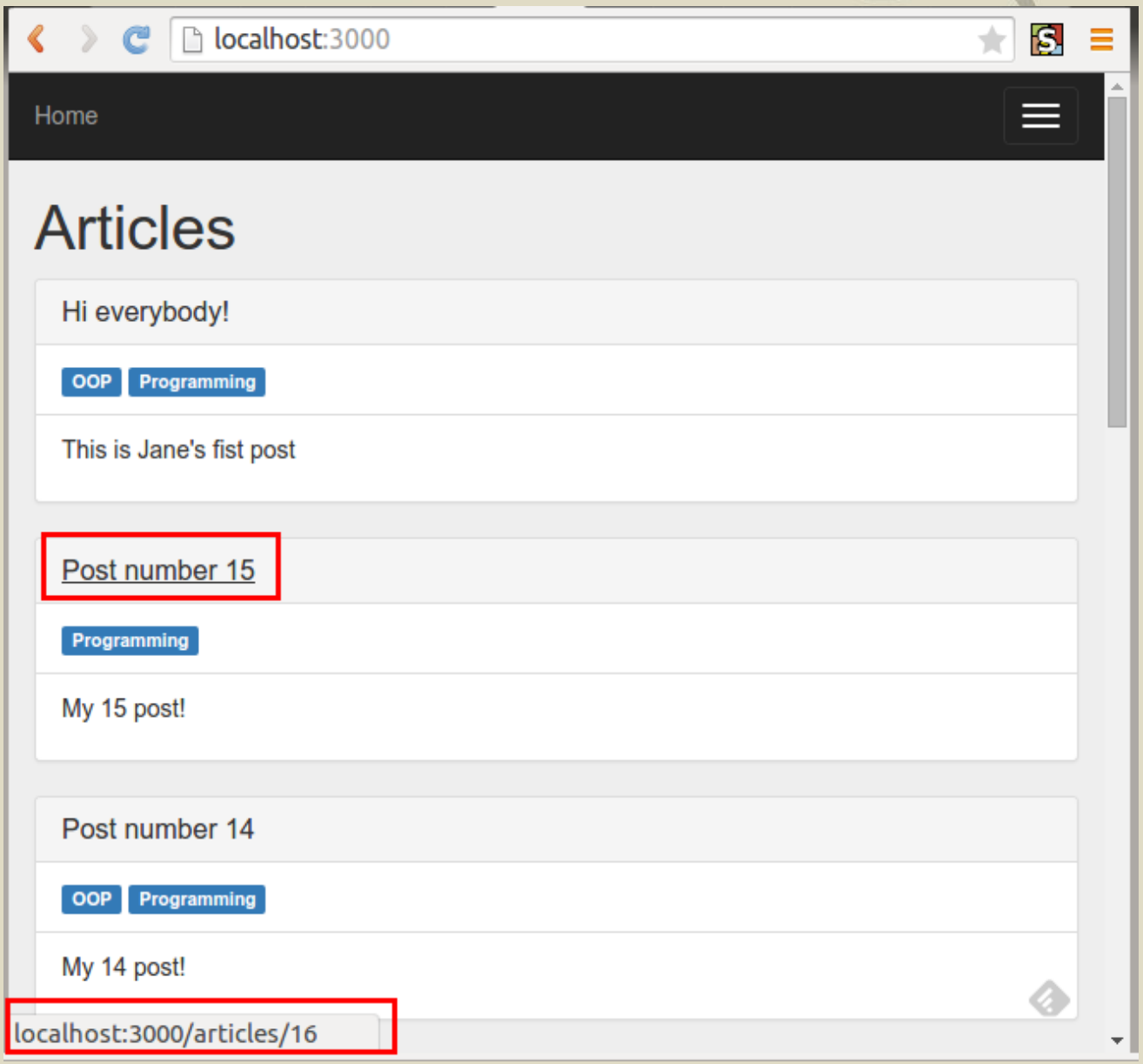
Repaso

- Entornos de Rails
  - *development (default).*
  - *test*
  - *production*
- Relaciones M:N usando join table
  - `has_and_belongs_to_many`

Relaciones M:N

Tarea para el hogar

- Modificar el índice para usar paneles de bootstrap.



Tarea para el hogar

- Ajustar los seeds.

Algunas *rake-tasks* de BDs

- db:migrate – Ejecuta migrations pendientes.
- db:create – Crea la BD.
- db:drop – Elimina la BD.
- db:schema:load – Crea el esquema en base a `schema.rb`.
- db:setup == db:create, db:schema:load, db:seed
- db:reset == db:drop, db:setup

Sobre modificar un *migration*

- Si modifican una migration
  - db:drop
  - db:create
  - db:migrate
- Con eso su esquema vuelve a estar alineado con las migrations.

Relaciones M:N

- Supongamos que queremos dar un orden a las categorías.
- Necesitamos agregar la información de la posición a la *join table*.

Relaciones M:N

articles

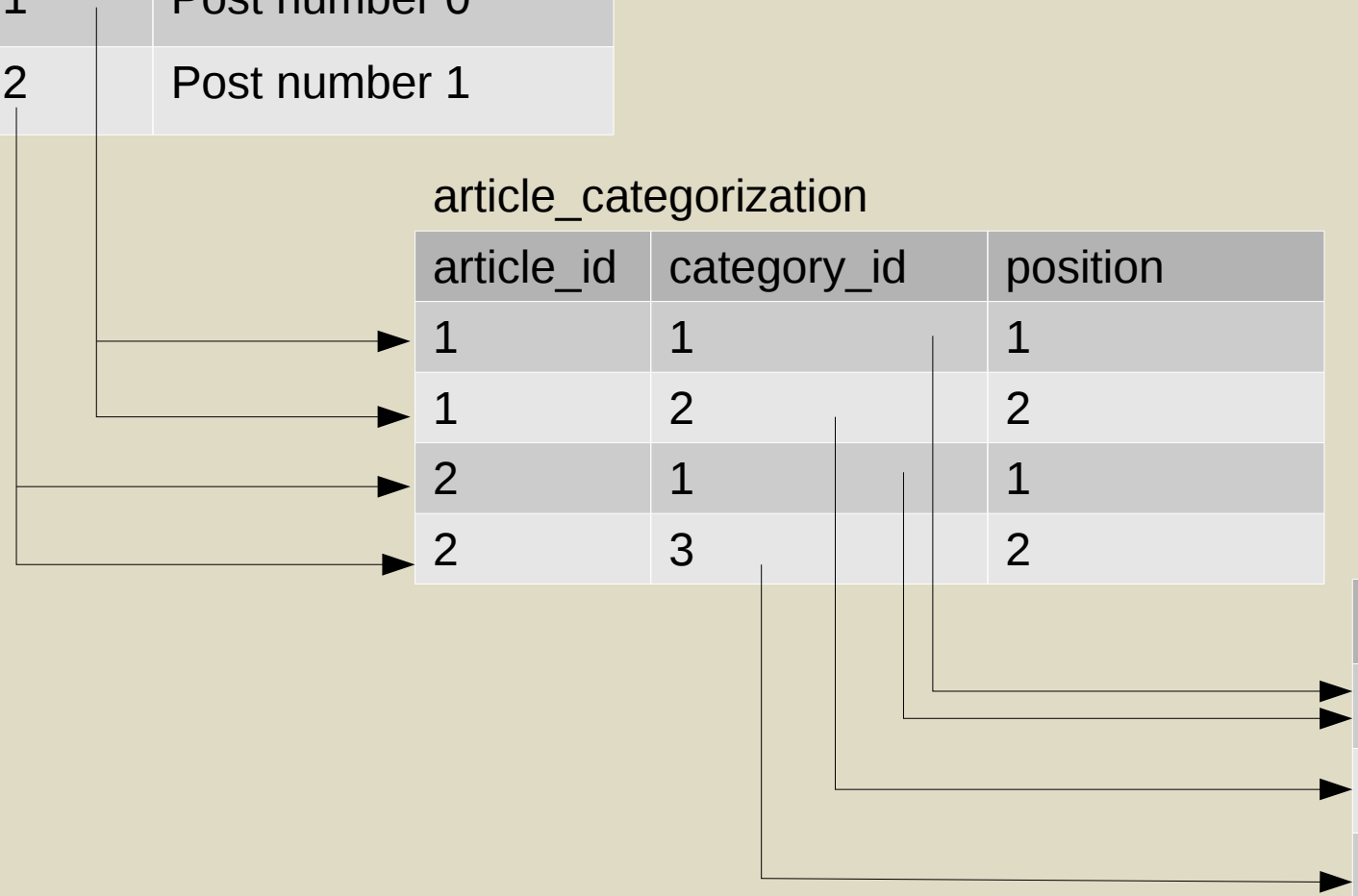
id	title
1	Post number 0
2	Post number 1

article_categorization

article_id	category_id	position
1	1	1
1	2	2
2	1	1
2	3	2

categories

id	name
1	OOP
2	Programming
3	Smalltalk



Relaciones M:N

- En Rails lo que hacemos es generar un *join model*.
  - Generamos un modelo.
  - Seteamos relaciones hacia las partes (`Article` y `Category`).
  - Le indicamos a ActiveRecord que ese modelo genera una relación a un tercer modelo.

Relaciones M:N

- Pasos
  - Crear un migration para eliminar la tabla actual.
  - Crear el *join model* y ejecutar la migration.
  - Configurar las relaciones.
- ¿Problemas?
  - En el primer paso perdimos los datos.
  - Tenemos que lograr convertir nuestra relación sin perder los datos.

Relaciones M:N

- Pasos
  - Crear el *join model* y ejecutar la migration.
  - Configurar las relaciones.
  - Migrar los datos.
  - Crear un migration para eliminar la tabla actual.

Relaciones M:N

- Comiteen cualquier cambio pendiente.
- Armen un nuevo branch.

Relaciones M:N

```
$ bin/rails g model ArticleCategorization article:references
category:references position:integer
```

/db/migrate/XYZ_create_article_categorizations.rb

```
class CreateArticleCategorizations <
ActiveRecord::Migration
  def change
    create_table :article_categorizations do |t|
      t.references :article, index: true
      t.references :category, index: true
      t.integer :position

      t.timestamps
    end
  end
end
```

Relaciones M:N

/app/models/article_categorization.rb

```
class ArticleCategorization < ActiveRecord::Base
  belongs_to :article
  belongs_to :category
end
```

Relaciones M:N

- Modifiquemos la definición de las relaciones en los modelos existentes

/app/models/category.rb

```
class Category < ActiveRecord::Base
 has_many :article_categorizations
 has_many :articles, :through => :article_categorizations
end
```

/app/models/article.rb

```
class Article < ActiveRecord::Base
 ...
 has_many :article_categorizations
 has_many :categories, :through=>:article_categorizations

end
```

Relaciones M:N

- Ejecutemos la migration

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

- Vayan al sitio (“normal” y admin).
- ¿Cómo recuperamos los datos de la join table para crear instancias de `ArticleCategorization`?

Relaciones M:N

- En una consola

```
$ bin/rails c
Loading development environment (Rails 4.1.8)
> ActiveRecord::Base.connection.select_all('select * from
articles_categories').each do |e|
 >     puts e.inspect
?>   end
(0.6ms)  select * from articles_categories
{"article_id"=>1, "category_id"=>1}
{"article_id"=>1, "category_id"=>3}
...
{"article_id"=>17, "category_id"=>1}
{"article_id"=>17, "category_id"=>2}
 => [{"article_id"=>1, "category_id"=>1}, ... ,
{"article_id"=>17, "category_id"=>2}]
```

Relaciones M:N

- Generemos una migration para los datos

```
$ bin/rails g migration migrate_categorization_data
```

/db/migrate/XYZ_migrate_categorization_data.rb

```
class MigrateCategorizationData < ActiveRecord::Migration
  def change
  end
end
```

Relaciones M:N

/db/migrate/XYZ_migrate_categorization_data.rb

```
class MigrateCategorizationData < ActiveRecord::Migration
  def up
    ActiveRecord::Base.connection.select_all('select *
from articles_categories').each do |article_category|
      article_id = article_category["article_id"]
      category_id = article_category["category_id"]
      article = Article.find(article_id)
      position = article.categories.count
      ArticleCategorization.create!(
        article_id: article_id,
        category_id: category_id,
        position: position)
    end
  end

end
```

Relaciones M:N

/db/migrate/XYZ_migrate_categorization_data.rb

```
class MigrateCategorizationData < ActiveRecord::Migration

  ...

  def down
    raise ActiveRecord::IrreversibleMigration
  end
end
```

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

- Verifiquen el sitio (normal y admin).

Relaciones M:N

- Ahora nos falta eliminar la vieja *join table*.

```
$ bin/rails g migration drop_articles_categories
```

/db/migrate/XYZ_drop_articles_categories.rb

```
class DropArticlesCategories < ActiveRecord::Migration
  def up
    drop_table :articles_categories
  end

  def down
    raise ActiveRecord::IrreversibleMigration
  end
end
```

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

Relaciones M:N

- Verifiquen el sitio
- Hagan un `db:reset` y verifiquen el sitio nuevamente

```
$ bin/rake db:reset
```

Relaciones M:N

- Algunas cosas a tener en cuenta
  - La posición de una categoría no puede ser nula.
  - No repetir el mismo par artículo – categoría.
  - Para un mismo par artículo – categoría no se puede repetir posición.
  - ¿Encapsular el manejo de categorías?

Relaciones M:N

- La posición de una categoría no puede ser nula.

/app/models/article_categorization.rb

```
class ArticleCategorization < ActiveRecord::Base
  belongs_to :article
  belongs_to :category

  validates_presence_of :article, :category, :position
end
```

- Intenten un `db:reset`.

Relaciones M:N

/app/models/article.rb

```
class Article < ActiveRecord::Base
  ...
  has_many :article_categorizations
  has_many :categories, :through =>
:article_categorizations

  def add_category(category)
    position = categories.count
    article_categorizations.create!(category: category,
position: position)
  end

end
```

- Modificar el seed.

Relaciones M:N

- Intenten un `db:reset` nuevamente.
- En una consola

```
$ bin/rails console
> ArticleCategorization.last
  ArticleCategorization Load (0.4ms)  SELECT
"article_categorizations".* FROM "article_categorizations"
ORDER BY "article_categorizations"."id" DESC LIMIT 1
 => #<ArticleCategorization id: 34, article_id: 17,
category_id: 2, position: 1, created_at: "2015-05-29
19:41:38", updated_at: "2015-05-29 19:41:38">
> ArticleCategorization.create!(article_id: 17, category_id:
3, position: 1)
```

Relaciones M:N

/app/models/article_categorization.rb

```
class ArticleCategorization < ActiveRecord::Base
 belongs_to :article
 belongs_to :category

 validates_presence_of :article, :category, :position

 validates_uniqueness_of :position, :scope => :article_id
end
```

Relaciones M:N

- `db:reset` nuevamente.
- En una consola

```
$ bin/rails console
> ArticleCategorization.create!(article_id: 17, category_id:
3, position: 1)
...
> ArticleCategorization.create!(article_id: 17, category_id:
3, position: 2)
> ArticleCategorization.create!(article_id: 17, category_id:
3, position: 3)
```

- Vayan al índice de artículos.

Relaciones M:N

- Validemos el par artículo - categoría

/app/models/article_categorization.rb

```
class ArticleCategorization < ActiveRecord::Base
 belongs_to :article
 belongs_to :category

 validates_presence_of :article, :category, :position

 validates_uniqueness_of :position, :scope => :article_id
 validates_uniqueness_of :category_id, :scope =>
:article_id
end
```