Working with the Database

"Active Record is the M in MVC - the model - which is the layer of the system responsible for representing business data and logic."

Migrations are used to make additions and changes to our database schema.

Table of Contents

ActiveRecord

ActiveRecord is covered in the Database + ActiveRecord = ? section of the notes.

Other ActiveRecord Resource:

- Active Record Query Interface Guide
- Active Record Validations Guide
- Active Record Associations Guide

Rails Migrations

“Migrations are a way to alter your database schema over time in a consistent way.

You can think of each migration as being a new ‘version’ of the database.

A schema starts off with nothing in it, and each migration modifies it to add or remove tables, columns, or entries.” – from the Rails Migration Guide

Model Generator Migrations

Model and scaffold generators will automatically create migrations appropriate for your new model.

```
rails g model Ghost name:string description:text
```

Will generate the following `ghosts` table migration:

```
class CreateGhosts < ActiveRecord::Migration[6.0]
  def change
    create_table :ghosts do |t|
      t.string :name
      t.text :description
      t.timestamps
    end
  end
end
```

Add Column Migration

Rails Migrations allow us to make incremental changes to our Models.

To add an `image` string property to an existing `products` table we use the command:

`rails g migration add_image_to_products image:string`

The naming here is important. Whenever we want to add a new column to a table the command follows this form:

```
rails g migration add_column_to_table column:type
```

(Where `column` in the name of the column to add, `table` is the name of the table to be modified, and `type` is the database of this column.)

The first migration command above would generate a migration in the `db/migrate` folder called `20121107184925_add_image_to_products.rb`:

```
class AddImageUrlToProducts < ActiveRecord::Migration
  def change
    add_column :products, :image, :string
  end
end
```

Next we use the rails command to run the migration:

```
rails db:migrate
```

Remove Column Migration

We can also use migrations to rename columns, remove columns, add indexes, etc.

For example, to remove a String column named `image` from the `products` table:

```
rails g migration remove_image_from_products image:string
```

The Schema File

When you run a migration your `db/schema.rb` is rewritten to match the up-to-date structure of your database.

If you ever want to rebuild your database from scratch but your migrations no longer run properly you can load the database directly from the schema file:

```
rails db:drop
rails db:schema:load
```

Other Helpful Migration Techniques

To best understand all that can be accomplished with migrations please read the Rails Guides page on Migrations.

Amongst other things the guide covers:

- Column Change Migrations
- Creating Join Tables
- Rolling Back to Undo Migrations
- Resetting the DB
