Models

But not that kind. But they can be ridiculously good looking.

Objectives

- Describe a models function in the MVC framework.
- Define Rails migration and use rake to make it happen.
- Implement migrations to update database tables.
- Operate ActiveRecord to save and access model data.

Agenda

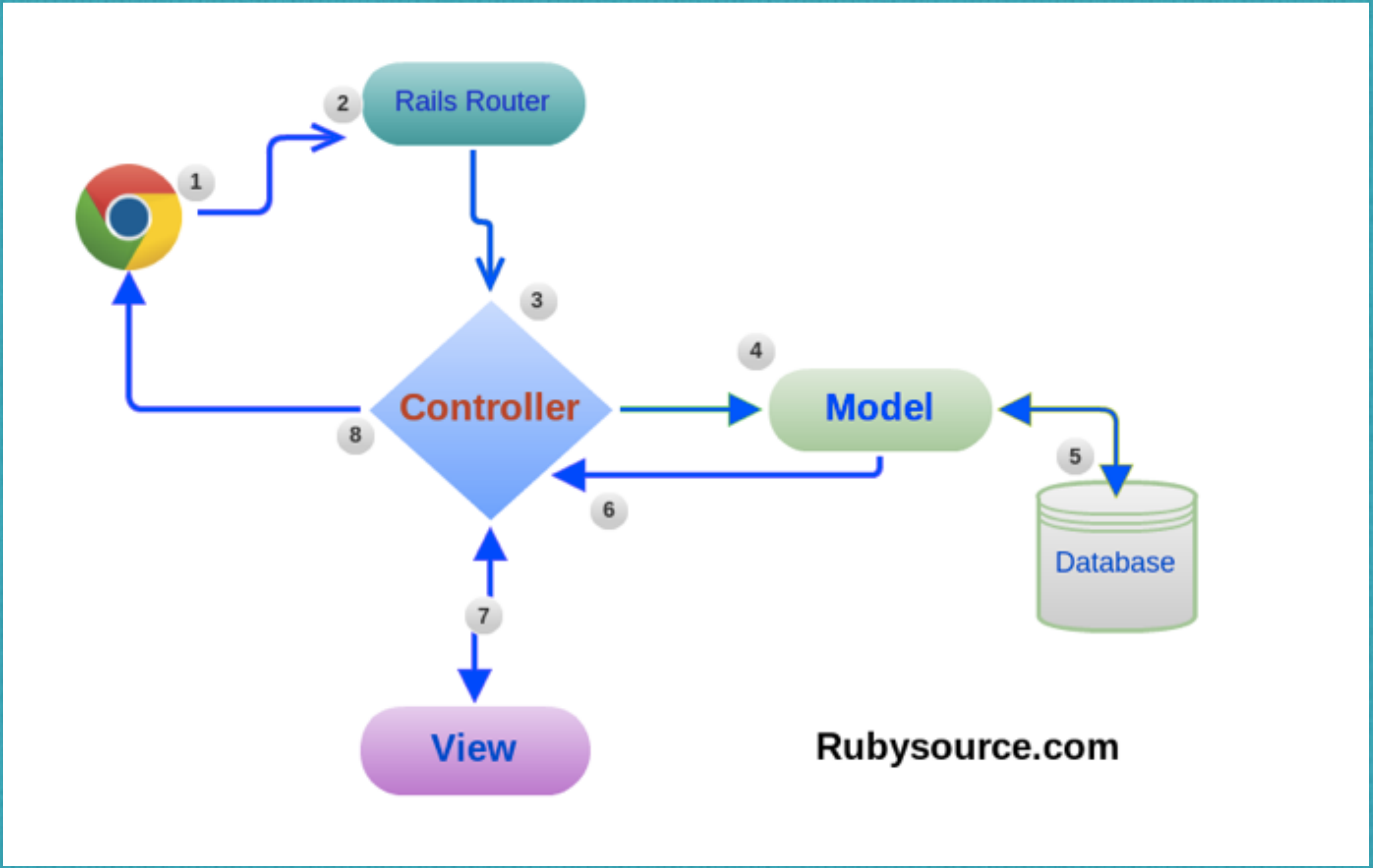
- ☐ Quick fire Movies app for review
- ☐ Models (The layer that talks to your database)
- ☐ Migrations (The tool that updates the structure of your DB)
- ☐ ActiveRecord and CRUD (Create, Read, Update, Delete)
- ☐ Lab Time - Movie App

Reviewing Controllers and Routes

- Let’s make a movies app

Movies App

- Repetition is important. Please don't get bored.
- Create a new application called movie_app in your class9 folder.
- Create a Movies controller with an index action.
- Set your root route to the index page.
  - IE: (http://localhost:3000/ goes to our movies index action)
  - http://guides.rubyonrails.org/routing.html and cmd + f if you get confused
  - Notice the lack of path after our domain, this is called the "root" route.
- `index.html.erb`, welcomes users to the movie app. (Welcome to RetFlix! See our collection of movies below).



Rails Router
1
2
3
Controller
4
Model
5
Database
6
7
8
View
Rubysource.com

Models: the last leg of the MVC table

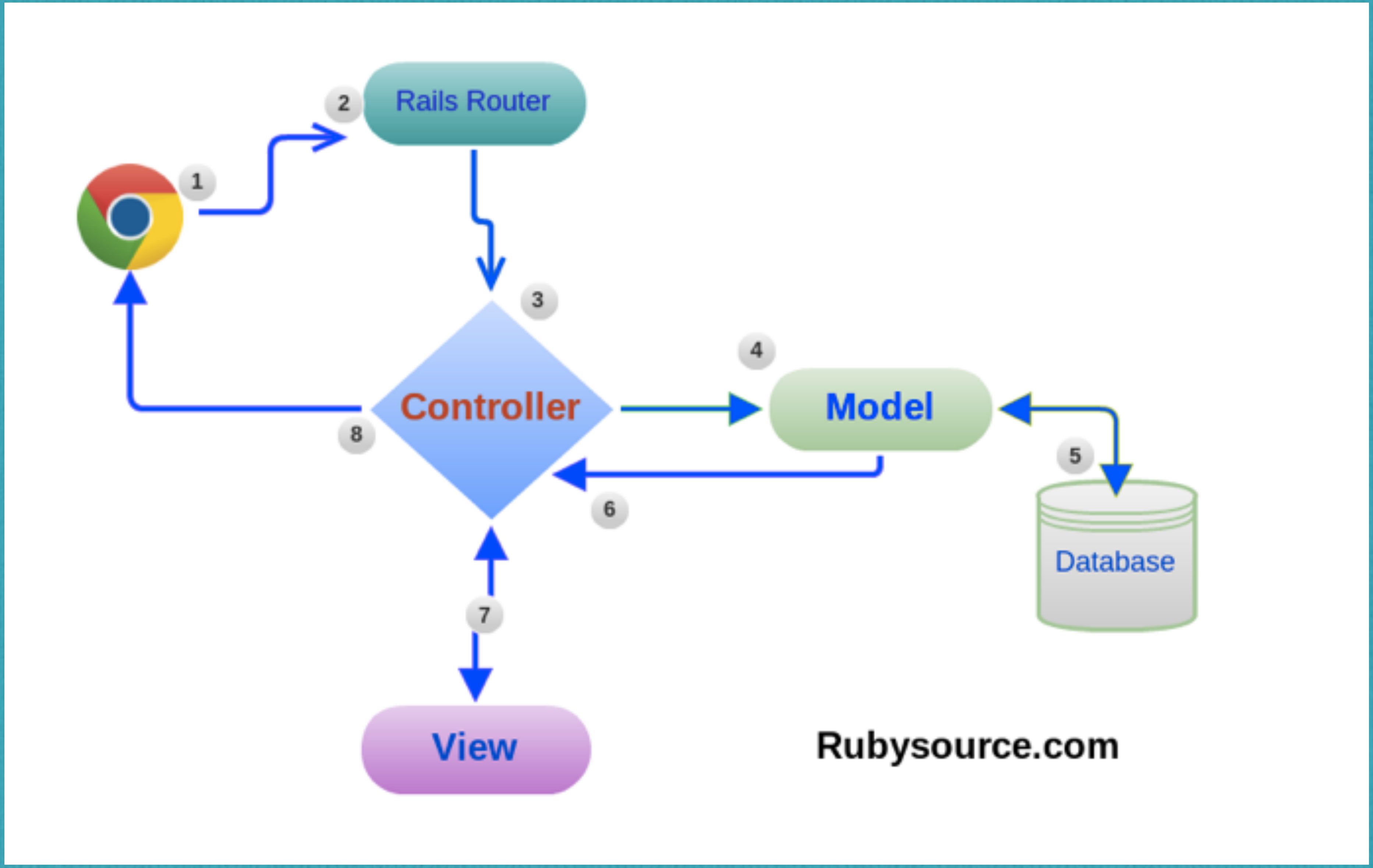
- Models shape the data that you store persistently
- They are Ruby classes
- That dictate the structure and behavior of database tables

The Flow

- The controller interacts with the Model
    - IE: Asks for data, creates data, updates, and deletes, <u>all with the model.</u>
    - The controller renders the view, potentially giving it data from a Model
    - Once the model gives back the data, the renders the view.
    - The view and the model do not interact (need the controller)

In a nut shell

- ☐ Controllers use models, and render views with the model data.
- ☐ Models store and retrieve information in a database
- ☐ Views only render markup, and nothing else!



The model and view are like soon-to-divorce parents that communicate via an unlucky offspring.

Models And the Database

- Models are required for interacting with our database
- We use a database to store data.
    - When we stop our rails server, the data will still exist in our database.
- Rails models make working with our database dead simple.
- Models are specific to a single table.
- Rails models have special functionality to allow you to easily lookup data from the table, or make changes without having to use SQL directly

Rails Models

- Always inherit from ActiveRecord::Base
- Always Capitalized and Singular
- They represent a single table in a database (but can associate to others)
- Models have attributes, but are defined by your database, not "`attr_accessor`"

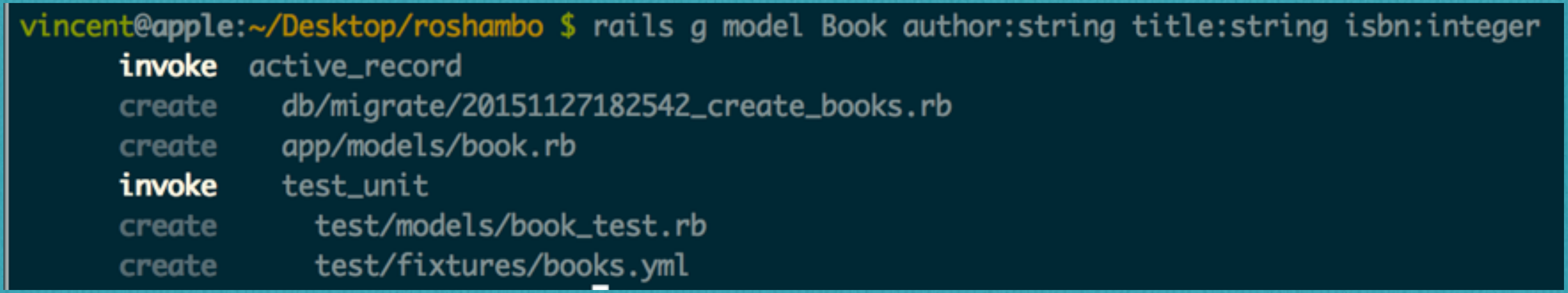
Generating models from the command line

It makes sense to let rails generate models for you.

Like so:

```
rails g model ModelName attribute_name:migration_type
attribute_2:migration_type
```

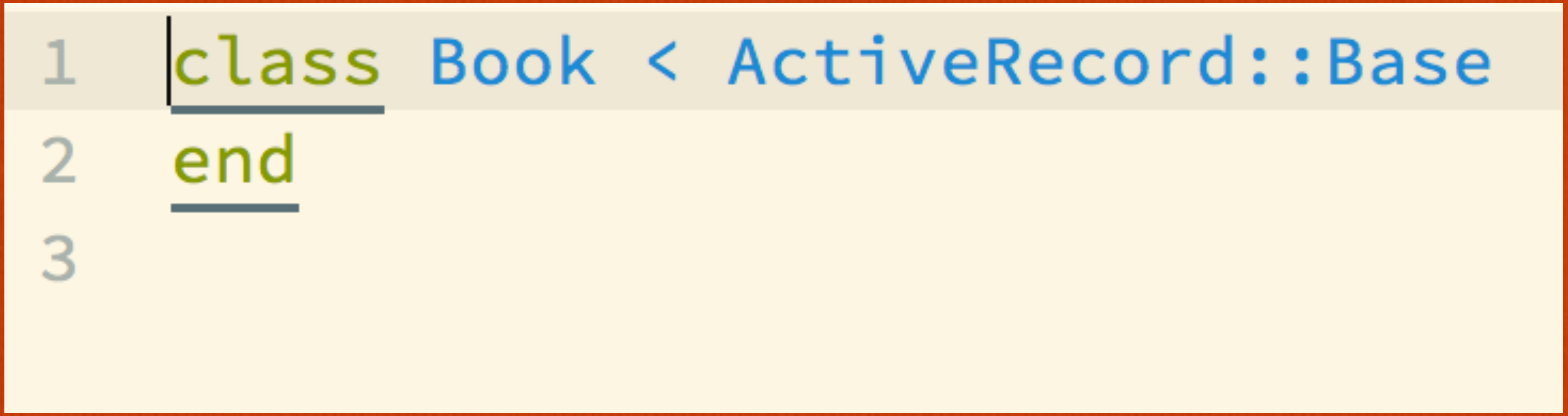
Let’s look at what that gave us



```
vincent@apple:~/Desktop/roshambo $ rails g model Book author:string title:string isbn:integer
      invoke  active_record
      create    db/migrate/20151127182542_create_books.rb
      create    app/models/book.rb
      invoke    test_unit
      create      test/models/book_test.rb
      create      test/fixtures/books.yml
```

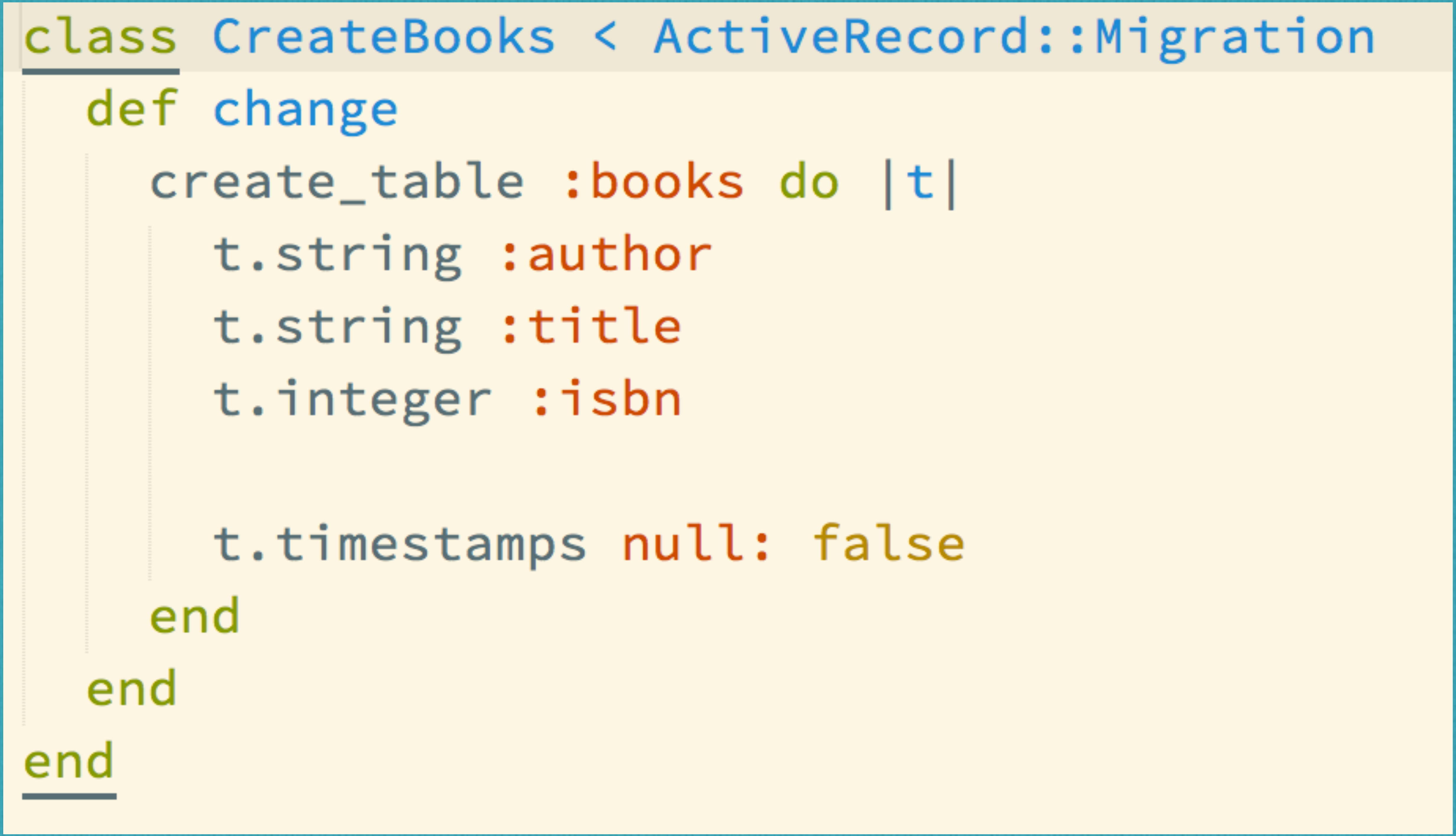
- A timestamped database migration file in db/migrate
- Model file in app/models
- Files for unit tests (we won’t cover that in this class, but definitely worth learning later)

Class for books inheriting from ActiveRecord::Base



```ruby
class Book < ActiveRecord::Base
end
```

Migration file, instructions for how the database will change. You can edit this but not after you migrate the database.



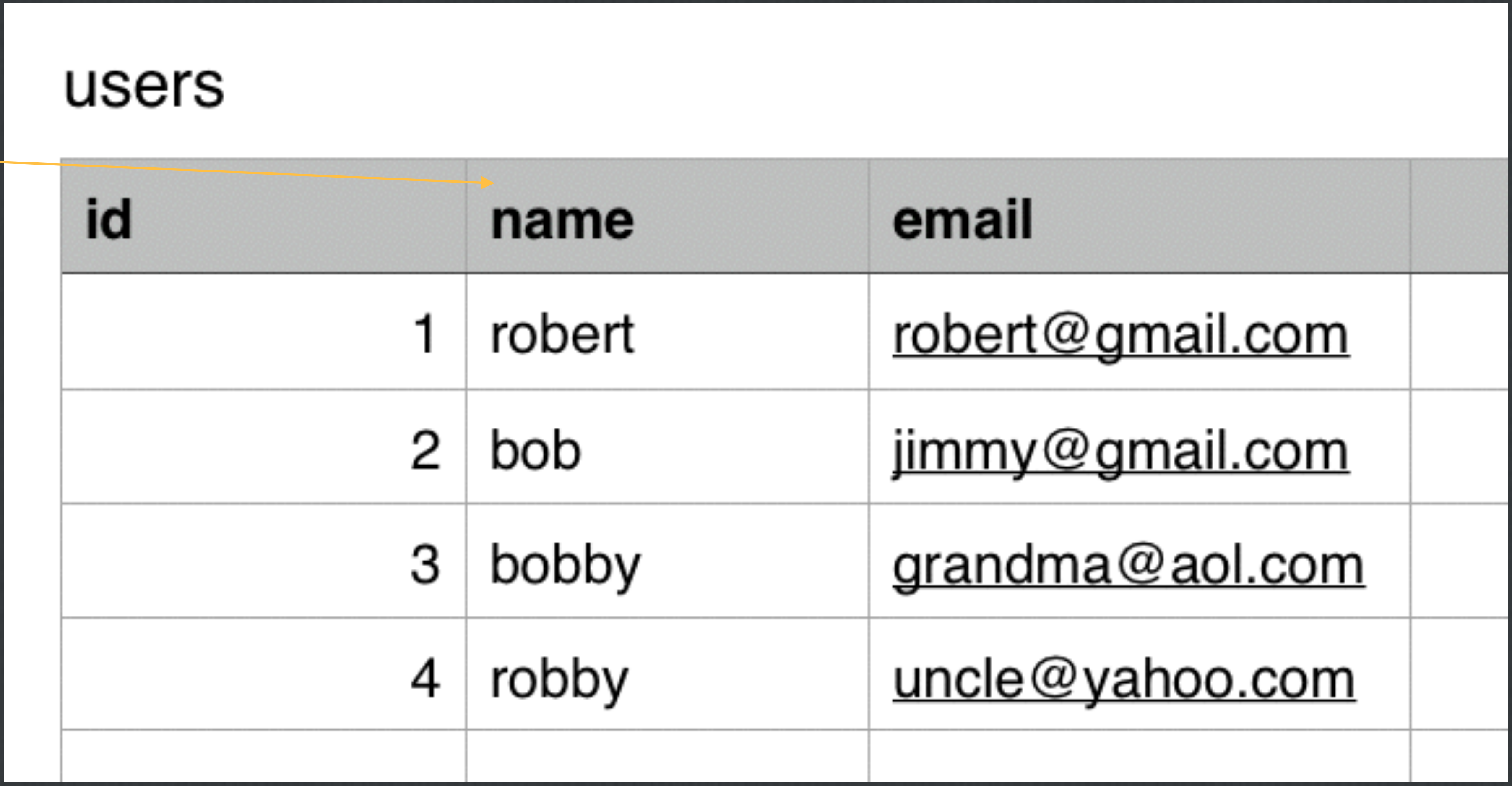
```ruby
class CreateBooks < ActiveRecord::Migration
  def change
    create_table :books do |t|
      t.string :author
      t.string :title
      t.integer :isbn

      t.timestamps null: false
    end
  end
end
```

Generating Models

- BE CAREFUL WHEN YOU TYPE THIS COMMAND — If you don’t think it through, you will have to do another migration to fix it.
- `rails g model Tshirt fabric:string color:string`
  - Creates a Tshirt model with 2 attributes, fabric and color
  - If you don’t specify a data type for a column (attribute) it defaults to string.
- rake db:migrate —> execute the migration file.
  - Creates the tshirts table in our database

Database defined attributes

attributes



users

id	name	email
1	robert	robert@gmail.com
2	bob	jimmy@gmail.com
3	bobby	grandma@aol.com
4	robby	uncle@yahoo.com

What is a database?

- Permanent store for data (lives beyond a single request to our server)
- Designed to handle data at scale
- Many different databases we can choose from, Rails handles almost all of them.
    - Prefer Postgres though in most situations, but we will use Rails’ default SQLite for most of the class.

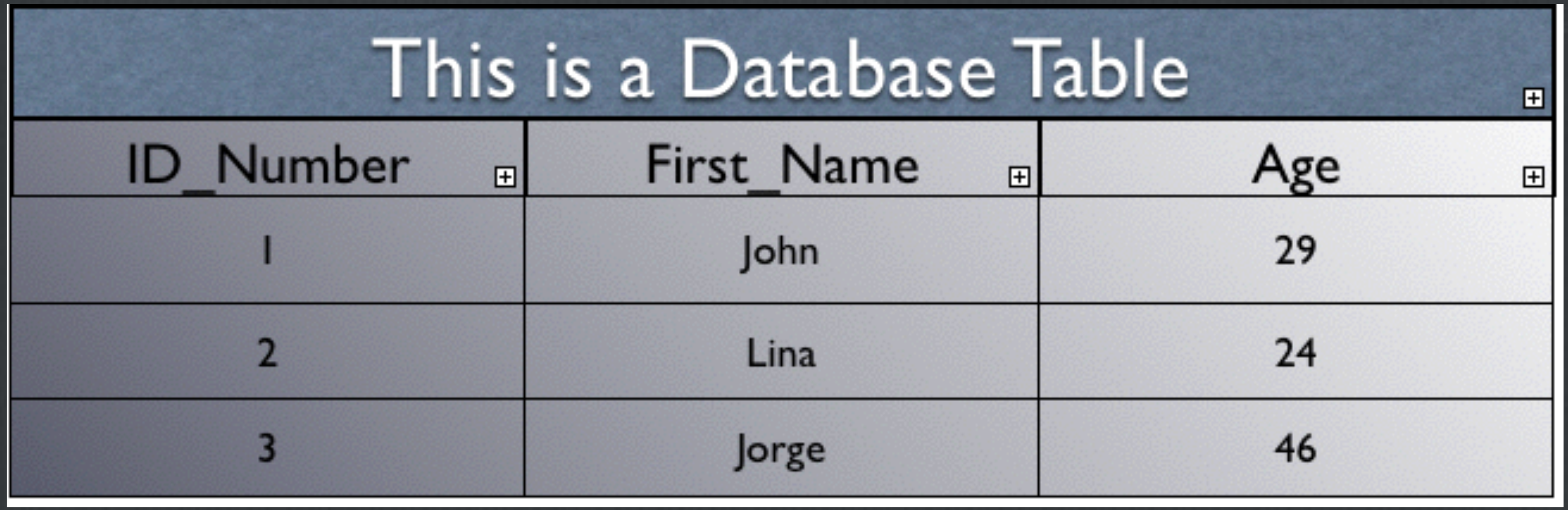
Some useful Data Types

:string
:text (longer strings)
:integer
:float
:boolean

Anatomy of a DB Table

- A Database will have multiple tables to represent multiple types of objects. (Users, Products, etc…)
- They have 2 big elements:
  - Rows — individual instances of the model
  - Columns — attributes specific to
- Most of the time, they will have an ID column (that is an ordinal number type, or sometimes a UUID).

For example



Columns →

Row →

Row →

Row →

This is a Database Table

ID_Number	First_Name	Age
1	John	29
2	Lina	24
3	Jorge	46

SQL

- SQL stands for “Structured Query Language”
- It is used to interact with databases.
- Looks like:

```
SELECT "movies".* FROM "movies" WHERE "movies"."title" =
'Jaws' LIMIT 1
```

- **Rails models generate SQL for us AND executes it**

For those of you who know SQL:

- You will not need to use it 99.9% (approx) of the time in Rails.
- Rails can generate almost every type of query you can think of for a database.

Why Migrations?

- ☐ They're good for developers coming into your app at a later time.
- ☐ It is a guaranteed way of ensuring your database is ALWAYS has the same structure.
- ☐ Even if you screw something up in a previous migration, you have to fix it with a new migration. Think of it like the 18th and 21st amendments to the Constitution. If we just erased the 18th, it erases part of who we are as a nation and how we got here.

Migrations

- You can edit the migration file BEFORE you migrate, but if you do it afterward, you will have to completely reset your database. Not good practice.
- Once you migrate, do further migrations (next class)
- Never ever ever ever edit schema.rb. Rake does that for you when you `rake db:migrate`

The Rails Console

- In your shirt management app, run rails c
- This opens the rails console, a live command line for your app.
- You can make manual changes to the database here.
- In the future, we will learn how to safely let users create, edit, and delete data through forms in the browser.

Code along

- Shirt Management app is an application we will build incrementally during class.
- The app allows users to manage their T-Shirts collection, by adding and deleting shirts to the database.
- For this lesson we will add a basic T-Shirt Model.
- Go ahead and `rails new shirt_management`
- `First story: display all shirts.`

Model methods

- You can initialize a new t-shirt with:
  - `abc = Tshirt.new(fabric: “Cotton”, color: “Red”)`
- You can save the t-shirt to the database with:
  - `abc.save`
- There’s a shortcut that does these both at the same time for you:
  - `Tshirt.create(fabric: “Cotton”, color: “Red”)`
- Same thing as .new and .save

Model methods

- You can get the first and last records with:
  - Tshirt.first, Tshirt.last
- You can get all t-shirts stored in the database with:
  - Tshirt.all
- The .all method returns something similar to an Array. (Which means you can use the .each method from a few classes ago!)

MOAR CODE

Shirt Management

Add workshirts to shirt management, just like we did for tshirts. Try to copy-paste as little as possible

15 minutes

Lab / Homework

Creating a new model

- Create a model in your movie app (the one from the beginning of class) called Movie with these attributes:
    - title
    - description
    - year_released
- Migrate the database

- ☐ Pair up. Use the seeds file to populate your database with movies.
- ☐ https://gist.github.com/trivett/88a0c92515e3cb378ebf
- ☐ Update your movie's index.html.erb so that it lists all movies in the database. Hint: (`@movies.each`)
- ☐ Make a show action for the individual movies.
- ☐ Rails console is also your friend. Use it to explore active record methods that will help you interact with the database.
- ☐ You should be able to go to http://localhost:3000/movies to see a list of all of the movies
- ☐ Note: This application is due lesson 11 (Dec. 12) — we will make it cooler

Homework

- Read: http://edgeguides.rubyonrails.org/active_record_basics.html
- Final project proposal