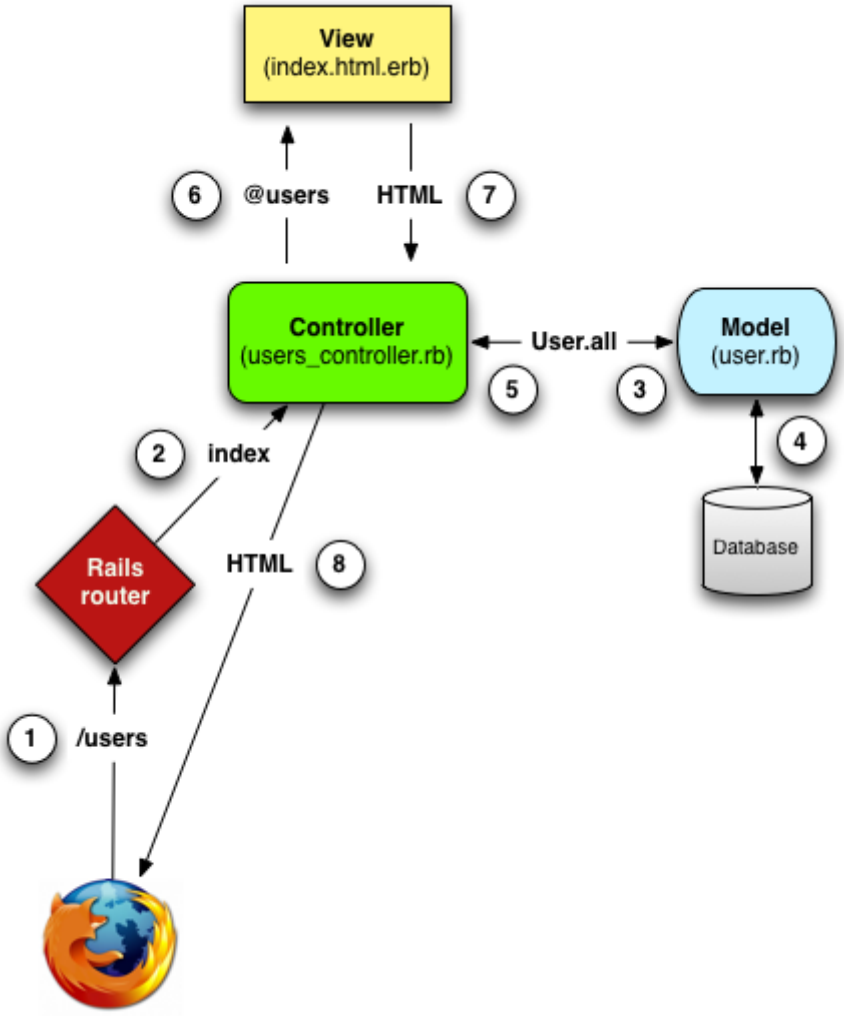
W5D4 - Models, Migrations, & Associations

Where we are in the course

- CSS throughout
- Ruby
  - OOP
- SQL
- Rails
  - Models <- we are here!
  - Controllers
  - Views
    - HTML
- JavaScript
  - JS Project
- React
- Redux
- Full-stack projects
- MERN Stack projects

- Jobsearch!!!

Rails: A server-side MVC web-application framework



ActiveRecord - an ORM Framework (The *M* in MVC)

Allows us to:

- Represent models and their data
- Represent associations between data
- Validate models before they get persisted to the database
- Perform database operations (**CRUD**) in OOP fashion

- *Migrations*
- Models
- Associations

Migrations

- *Incremental* and *reversible* changes made to a database schema, allowing it to evolve over time.
- Not just a Rails thing - ubiquitous to app frameworks that work with relational DBs.
- Rails allows you to use an easy Ruby DSL (domain-specific language) to describe changes to your tables, rather than write raw SQL.

Let's Migrate!

Common migration terminal commands

- `bundle exec rails g migration Create{TableName}`
- `bundle exec rails g migration Add{ColumnName}To{TableName}`
- `bundle exec rails g migration Remove{ColumnName}From{TableName}`
- `bundle exec rails g migration AddIndexTo{TableName}`

Common migration methods

- `create_table`
- `add_column`
- `change_column`
- `add_index`

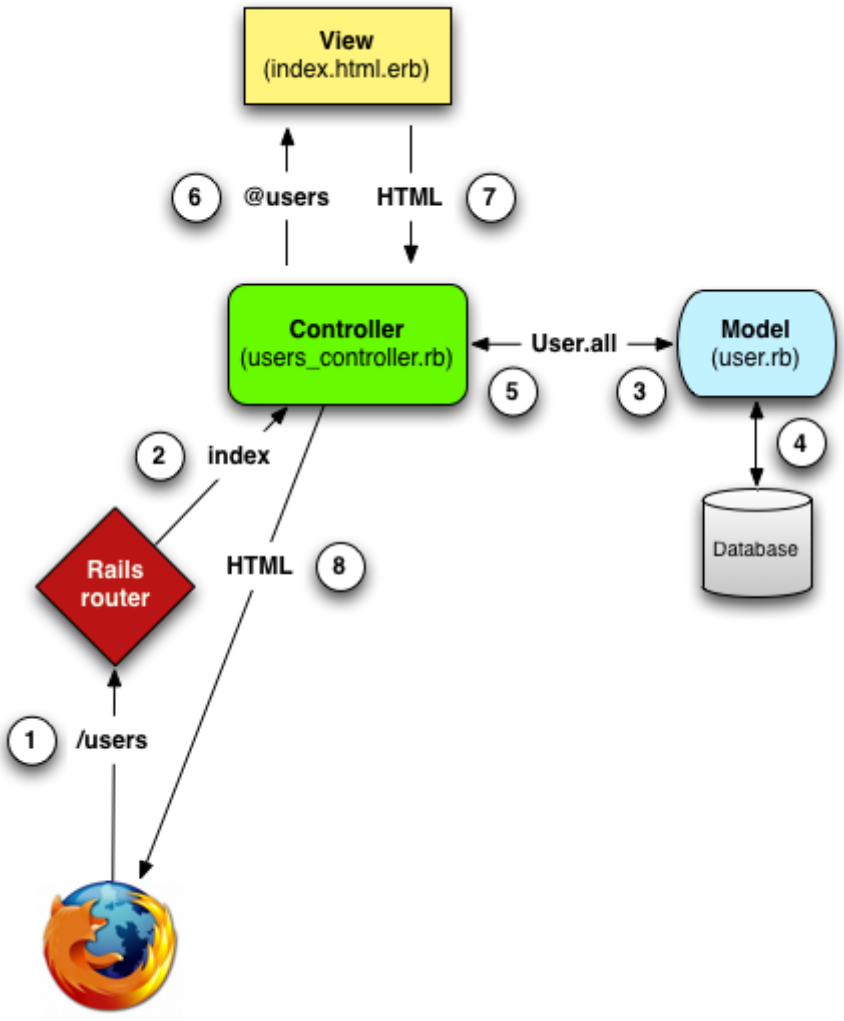
Changing existing migrations

- You can't just edit the migration and run the migration again
- Instead, you have two options:
    i. Write a new migration (much preferred)
    ii. Rollback:
        - rollback the migration (via `rails db:rollback`)
        - *then* edit your migration
        - run `rails db:migrate` to run the corrected version.

- ~~Migrations~~
- *Models*
- Associations

Model

- The central component of the MVC pattern
- A class that represents and directly manages the data, logic, and rules for a table
  - Typically contains: validations, associations, and custom methods
  - Inherits from ApplicationRecord (which, in turn, inherits from ActiveRecord)
- There is a one-to-one correspondence between a model and a table
- An instance of this class / model represents a row in our table



Database Constraints vs Model Validations

- Model validations are best used to provide error messages to users interacting with your app
- It's highly likely that:
  - You will interact with the database at some point outside of Rails
  - You will make a mistake in your code that causes invalid data
- Database constraints are the last line of defense for data-integrity

Common Validations

- `validates :some_column, presence: true`
  - similar to `null: false`
- `validates :other_column, uniqueness: true`
  - similar to `unique: true`
- Custom Validations

Rails Models Demo

- ~~Migrations~~
- ~~Models~~
- *Associations*

Associations

- Connections between two Active Record models.
- Make common operations simpler and easier in your code.
- We don't have to write anymore SQL `JOIN` statements
- Simply methods that we can call

Example Association

```
class Strike < ApplicationRecord
  belongs_to :student,
        primary_key: :id,
        foreign_key: :student_id,
        class_name: :Student
end
```

- belongs_to is an ActiveRecord method that takes the following arguments

```
def belongs_to(:name, options = {})
end
```

Associations Code Demo

Associations Recap

```
class Post < ApplicationRecord
  #validations go here
  belongs_to :user,
    primary_key: :id,
    foreign_key: :author_id,
    class_name: :User
end

class User < ApplicationRecord
  #validations go here
  has_many :posts,
    primary_key: :id,
    foreign_key: :author_id,
    class_name: :Post
end
```

has_one

- easily confused with `belongs_to`

- only write them if you've already made the corresponding belongs_to

Strategy when writing associations

- Start with `belongs_to`
- Write the corresponding `has_many` or `has_one` in the other model.
- Write `has_many` throughs using **only** other associations in the model as the `through`, check the associated model for an association to be the source.

Terminal Commands Recap

- `rails _5.2.3_ new {project_name} -G --database=postgresql --skip-turbolinks`
- `bundle exec rails db:create`
- `bundle exec rails g migration Create{TableName}`
- `bundle exec rails db:migrate`
- `bundle exec rails db:migrate:status`
- `bundle exec rails g model {ModelName}`
  - (creates migration file and model file)

Thank you!