Rails Guide

Hey, thanks a lot for picking up this guide!

I created this guide as a quick reference for when you are working on your projects, so you can quickly find what you need & keep going.

Hope it helps you!

MVC Architecture

Letter	Full Name	Description
M	Model	Everything database related.
V	View	The front-end / HTML / CSS, what the user of your app sees.
C	Controller	Glues the Model & the View.

Migrations

System to update your database schema.

Example 1:

```
rails generate migration AddPartNumberToProducts part_number:string
```

Example 2:

```
rails generate migration AddColumnToPosts user:references
```

Example 3:

```
rails generate migration CreateUsers name:string email:string
```

Types of values:

- string (less then 255 characters)
- text

- datetime
- timestamp
- float
- decimal
- integer
- boolean
- references

Run migrations:

```
rake db:migrate
```

Migration Formats:

```
CreateXXX        -> create_table
AddXXXToYYY      -> add_column
RemoveXXXFromYYY -> remove_column
```

Routes

Is the part of Rails that knows how to handle URLs & match them to a controller.

Set the main page for your website:

```
root "pages#index"
```

Create new routes:

```
get  "/users", to: "users#index", as: "users"
get  "/users/:id", to: "users#show", as: "show_user"
post "/users/create", to:"users#create", as: "create_user"
```

Create a set of related routes:

```
resources :users  # Has index route, plural
resource :post    # No index route, singular
```

Listing Routes:

```
rake routes
```

Scaffolding

Create MVC files, migrations & testing templates.

Example:

```
rails generate scaffold Post votes:integer link:text
```

ERB

Embbeded Ruby. Allows you to have Ruby code inside HTML.

How to embbed Ruby code:

```
<% 123 %>   # Doesn't show output on page
<%= 123 %>  # Shows output on page
```

Loop:

```
<% @posts.each do |post| %>
  <p><%= post.votes %></p>
  <p><%= post.link %></p>
<% end %>
```

Links:

```
<%= link_to 'Upvote', upvote_post_path(post), method: 'post' %>
<%= link_to 'Edit', edit_post_path(post) %>
<%= link_to 'Show', post %>
```

Validations

Allow you do define what data is valid & what is not valid.

Example:

```
validates :votes, presence: true
validates :link, uniqueness: true
```

Format:

```
validates :legacy_code,
    format: { with: /\A[a-zA-Z]+\z/, message: "only allows letters" }
```

Inclusion:

```
validates :size, inclusion: { in: %w(small medium large),
    message: "%{value} is not a valid size" }
```

Length:

```
validates :name, length: { minimum: 2 }
validates :bio, length: { maximum: 500 }
validates :registration_number, length: { is: 6 }
```

Numericality:

```
validates :points, numericality: true
validates :age, numericality: { greater_than: 21 }
validates :votes, numericality: { less_than: 10 }
```

Custom Validation

With validation class:

```
class EmailValidator < ActiveModel::EachValidator
  def validate_each(record, attribute, value)
    unless value =~ /\A([^@\s]+)@((?:[-a-z0-9]+\.)+[a-z]{2,})\z/i
      record.errors[attribute] << (options[:message] || "is not an email")
    end
  end
end

class Person < ApplicationRecord
  validates :email, presence: true, email: true
end
```

With validation method:

```
validate :expiration_date_cannot_be_in_the_past

def expiration_date_cannot_be_in_the_past
  if expiration_date.present? && expiration_date < Date.today
    errors.add(:expiration_date, "can't be in the past")
  end
end
```

Conditional Validation

```
class Order < ApplicationRecord
  validates :card_number, presence: true, if: :paid_with_card?

  def paid_with_card?
    payment_type == "card"
  end
end
```

ActiveRecord

Query the database, update & delete data.

Query:

```
User.all
User.first
User.find_by(email: "test@example.com")
User.where("id > ?", params[:min_id])
```

Updating:

```
# Find one user
@user = User.first

# Update attributes
@user.name = "John"

# Save changes
@user.save
```

Rails Naming Conventions

```
Model          - Singular
Database table - Plural + snake_case
Controller     - Plural
Views Folder   - Plural
```

Example:

```
Model        - User
Table        - user
Controller   - UsersController
Views Folder - users
```

Heroku Deploy

When you are ready to deploy your application to heroku you will need to make sure you have created an account & installed the heroku CLI as per the instructions.

Then you will need to login & create a new app.

Login:

```
heroku login
```

Create new app:

```
heroku create
```

Push changes:

```
git push heroku master
```

Read logs:

```
heroku logs
```

Run migrations:

```
heroku exec rake db:migrate
```

Also make sure that you have the `pg` gem on your `Gemfile` & that `config/database.yml` is setup to use postgres in production.

You also want to use postgres locally if possible to avoid any incompatible changes.

Controller Actions

Share data with views via instance variables.

Example:

```
def index
  @users = Users.all
end
```

Redirect to another action:

```
def upvote
  redirect_to action: 'index'
end
```

Layouts

If you want to create a menu or have some elements that show on all of the pages of your site then you need to use layouts.

By default your main layout file is under `app/views/layouts/application.html.erb`.

It's just a regular view file that you can edit & customize to your needs.

Associations

3 types:

- One-to-One

- One-to-Many
- Many-to-Many

Example:

```
class User < ApplicationRecord
  has_many :posts
end

class Post < ApplicationRecord
  belongs_to :user
end
```

Make sure that `Post` has a `user_id` column:

```
rails generate migration AddColumnToPosts user:references
```

Then:

```
rake db:migrate
```

To get all the posts for a user:

```
u = User.first

u.posts
```

Thanks for reading this guide!

- Jesus Castello (www.rubyguides.com)