RAILS ESSENTIAL

**Helpful Bookmarks**

* [Rails Routing from the Outside In](https://guides.rubyonrails.org/routing.html)
* [link\_to](https://apidock.com/rails/ActionView/Helpers/UrlHelper/link_to)
* [Action View Form Helpers (using form\_for)](https://guides.rubyonrails.org/v5.2/form_helpers.html)
  + [form\_for (ActionView::Helpers::FormHelper) - APIdock](https://apidock.com/rails/ActionView/Helpers/FormHelper/form_for) - form\_for is the older way of creating a form in Rails *with* an ActiveRecord model; it has been 'soft deprecated' and will be removed at some point in future versions of Rails
  + [form\_tag (ActionView::Helpers::FormHelper) - APIdock](https://apidock.com/rails/ActionView/Helpers/FormTagHelper/form_tag) - form\_tag is the older way of creating a generic form in Rails *without* an ActiveRecord model; it has been 'soft deprecated' and will be removed at some point in future versions of Rails
* [Strong Params](https://guides.rubyonrails.org/action_controller_overview.html#strong-parameters)

**Rails Commands**

* rails new <app-name> - create a new Rails app
* rails c - open a console
* rails s - start server
* rails routes - display all the routes in your app, also can viewed at <http://localhost:3000/rails/info/routes>
* rails g migration <migration-name> <attribute:data-type> <attribute:data-type> - generate a migration
* rails g model <model-name> <attribute:data-type> <attribute:data-type> - generate a model and a migration to create the table with specified columns
* rails db:migrate - run all pending migrations (same as rake db:migrate in Mod 1/Sinatra)

**Generators**

Rails generators help make developing Rails apps faster by writing a lot of boilerplate code for you. You should get comfortable knowing which generator to use based on what specific problem you're trying to solve in your application. Here are some commonly used generators:

* rails g migration: Create a migration file, similar to rails db:create\_migration
  + ex: rails g migration create\_student name age:integer
* rails g model: Creates a model with any attributes you specify, and a migration for that model. Model names must be *singular*.
  + ex: rails g model Student name age:integer
* rails g controller: Create a controller and any methods you specify, along with the corresponding routes and views. Controller names should be *plural*.
  + ex: rails g controller students index show
  + **warning** the controller generator will generate view files for anything that it is given as an additional argument, so rails g controller students new create destroy will make view templates for the new, create, and destroy methods (even though you probably don't want views for them)
* rails g resource: Create the model, migration, and controller for a given model. Resource name should be *singular*.
  + ex: rails g resource Student name age:integer
* rails g scaffold: don't use this! It generates a lot of boilerplate code that you won't need, and makes too many assumptions about how we want our app to work.

If you ever need to 'undo' a generator and remove all the files it created, you can also run the destroy command (ex. rails d model student).

More on generators: [The Rails Command Line — Ruby on Rails Guides](https://guides.rubyonrails.org/command_line.html#rails-generate)

**Routes**

There are two ways we primarily define routes:

* Using resources: Rails will automatically create the RESTful routes for a given model.
  + ex. resources :students, only: [:index, show] will create the following routes:
    - get '/students', to: 'students#index', as: 'students'
    - get '/students/:id', to: 'students#show', as: 'student'
* Custom routes: You need to specify the HTTP verb + path, controller + method, and optionally the path helper. These are helpful if you need to create routes outside of the seven RESTful routes provided by resources.
  + ex: patch /students/:id/toggle, to: 'students#toggle', as: 'toggle\_student'

More on routes: [Rails Routing from the Outside In — Ruby on Rails Guides](https://guides.rubyonrails.org/routing.html)

**Action View helpers**

Action View is one of the core libraries in Rails. It comes with a lot of helper methods that make it easier to write HTML strings. Some examples you'll see frequently:

* link\_to: generates an <a> (anchor) tag

<%= link\_to 'Students', students\_path %>

# => "<a href="/students">Students</a>"

* button\_to: generates a <form> tag with a submit button

<%= button\_to 'Delete', @student, method: :delete %>

# => "<form class="button\_to" method="post" action="/students/1">

# <input type="hidden" name="\_method" value="delete">

# <input type="submit" value="Delete">

# <input type="hidden" name="authenticity\_token" value="...">

# </form>"

* form\_for: helps generate HTML forms, useful for working with models. For an edit form, use an existing instance of a model; for a create form, use a new instance of a model (@student = Student.new). Based on what is passed as the first argument to form\_for, it will generate the correct method and action in the <form> tag.

<%= form\_for @student do |f| %>

<%= f.text\_field :name %>

<%= f.number\_field :age %>

<%= f.submit %>

<% end %>

# => "<form class="new\_student" id="new\_student" action="/students" method="post">

# <input type="hidden" name="authenticity\_token" value="...">

# <input type="text" name="student[name]" id="student\_name">

# <input type="number" name="student[age]" id="student\_age">

# <input type="submit" name="commit" value="Create Student">

# </form>"

* form\_tag: helps generate HTML forms; useful when you're not working with a model instance (for example, a search form). More generic than the form\_for helper.

<%= form\_tag "/students/search", method: :get do %>

<%= text\_field\_tag :search %>

<%= submit\_tag "Search" %>

<% end %>

# => "<form method="get" action="/movies">

# <input type="text" name="search" id="search">

# <input type="submit" name="commit" value="Search">

# </form>"

Link To/Button To: [ActionView::Helpers::UrlHelper](https://api.rubyonrails.org/v5.2.3/classes/ActionView/Helpers/UrlHelper.html#method-i-link_to) Form For: [form\_for (ActionView::Helpers::FormHelper) - APIdock](https://apidock.com/rails/ActionView/Helpers/FormHelper/form_for)

**Strong Params**

Rails does not allow mass assignment by default. If you want to use mass assignment to create/update an instance of your model, you have to whitelist specific params them using by calling the permit method on the params hash and passing in the specific keys you want to permit. This protects us from malicious users changing the names on input fields to try and update fields in our models we don't want them to update.

Typically, this is how you will see strong params used in a controller:

class StudentsController < ApplicationController

def create

# call the student params method to return the whitelisted params

student = Student.create(student\_params)

redirect\_to student\_path(student)

end

# private methods can only be called from within the same class

private

def student\_params

# params.require: will throw an error if the key isn't present in the params hash

# params.permit: whitelist these keys for mass assignment

params.require(:student).permit(:name, :age, :favorite\_food)

end

end