RAILS

Rails (1/3)

- Presentation
- Model-View-Controller Architecture
- Routing
- Controller
- Model
- View

Presentation

Ruby On Rails (RoR), or Rails, is a web application development **framework** written in Ruby and created in 2003 by David Heinemeier Hansson (Basecamp).
-- https://guides.rubyonrails.org/getting_started.html

- Written in Ruby
- Do not Repeat Yourself (**DRY**) vs WET
- **Convention Over Configuration** (more about that)
- Last version is 6.0 (we will use 5.2)
- Developer Toolbox (*Scaffolding*, *HTTP Server*, Rake, ...)
- Other web Framework (Symphony, Larevel, Sinatra, Express, Django)
- Used by Shopify, *Github*, AirBNB, Kickstarter, Soundcloud, ...

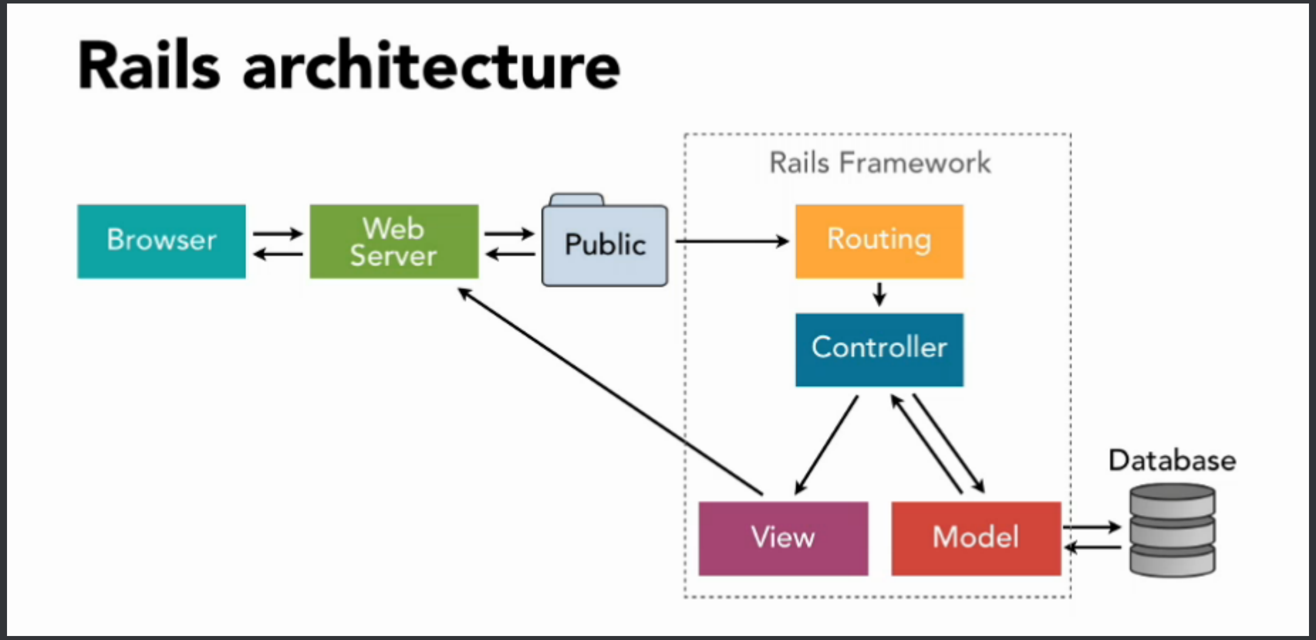
Convention Over Configuration

"Convention over Configuration" means a developer only needs to specify unconventional aspects of the application.
For example, if there is a class Sale in the model, the corresponding table in the database is called sales by default. It is only if one deviates from this convention, such as calling the table "products sold", that the developer needs to write code regarding these names.
Generally, Ruby on Rails conventions lead to less code and less repetition

⚠️ If you fight the framework, you're probably doing it wrong ⚠️

Model-View-Controller Architecture

This provides a clean isolation among the business logic in the Model, the user interface through the Views, as well as the processors handling all sorts of user requests in the Controller. This also makes for easier code maintenance.



-- https://medium.com/the-renaissance-developer/ruby-on-rails-http-mvc-and-routes-f02215a46a84

Routing

The Rails router recognizes URLs and dispatches them to a controller's action, or to a Rack application. It can also generate paths and URLs, avoiding the need to hardcode strings in your views.
— https://guides.rubyonrails.org/routing.html

- Rails routes are store in **`config/routes.yml`**
- A route allow a specific HTTP Request to call the correct *controller*

-- https://guides.rubyonrails.org/routing.html

```ruby
# config/routes
Rails.application.routes.draw do
  resources :messages
end
```

HTTP Verb	**Path**	**Controller#Action**	**Used For**
GET	/messages	messages#index	return a list of all messages
POST	/messages	messages#create	create a new message
GET	/messages/:id	messages#show	fetch a specific message
PATCH/PUT	/messages/:id	messages#update	update a specific message
DELETE	/messages/:id	messages#destroy	delete a specific message

-- https://www.codecademy.com/articles/standard-controller-actions

(Action) Controller

- Action Controller is the C in MVC.
- After the router has determined which controller to use for a request, the controller is responsible for making sense of the request, and producing the appropriate output.
- For most conventional RESTful applications, the controller will receive the request (this is invisible to you as the developer), fetch or save data from a model, and use a view to create HTML/JSON output
- A controller can thus be thought of as a middleman between models and views. It makes the model data available to the view so it can display that data to the user, and it saves or updates user data to the model.

-- https://guides.rubyonrails.org/action_controller_overview.html

```ruby
# Subclass of ApplicationController
class MessagesController < ApplicationController
  # GET /messages
  def index
    # messages = ... # fetch all message
    render json: messages
    # { id: 123, text: 'Hello'}, { id: 456, text: 'World'}
  end

  # POST /messages
  def create
    message = ... # create message
    render json: message, status: 201
  end

  # GET /messages/:id
  def show
    message = ... # find message
    render json: message
  end

  # PUT /messages/:id
  def update
    message = ... # find/update message
    render json: message
  end

  # DELETE /messages/:id
  def destroy
    # find/delete message
    head :no_content # return 204
  end
end
```

Model

- Active Record is the M in MVC - the model - which is the layer of the system responsible for representing business data and logic.
- Active Record facilitates the creation and use of business objects whose data requires persistent storage to a database.
- It is an implementation of the Active Record pattern which itself is a description of an Object Relational Mapping system.

-- https://guides.rubyonrails.org/active*record*basics.html

Model / Class	**Table / Schema**
Messages	messages
Chats	chats

```ruby
# subclass ApplicationRecord
class Messages < ApplicationRecord
end
```

```ruby
class MessagesController < ApplicationController
  # GET /messages
  def index
    messages = Messages.all
    render json: messages
  end

  # POST /messages
  def create
    message = Message.create!(text: params[:text])
    render json: message, status: 201
  end

  # GET /messages/:id
  def show
    message = Message.find(params[:id])
    render json: message
  end

  # PUT /messages/:id
  def update
    message = Message.find(params[:id])
    message.update!(text: params[:text])
    render json: message
  end

  # DELETE /messages/:id
  def destroy
    message = Message.find(params[:id])
    message.destroy!
    head :no_content # return 204
  end
end
```

View

- A View is the V in MVC - the view (!) - which is the layer of the system responsible for presenting the data.
- In the case of an API only Rails Application, it can be as simple as letting Rails convert the model itself

-- https://api.rubyonrails.org/v5.2/classes/ActiveModel/Serializers/JSON.html

```
class MessagesController < ApplicationController
  # ...

  # GET /messages/:id
  def show
    message = Message.find(params[:id])
    render json: message.as_json(only: [:id, :text])
  end

  # ...
end
```