CHAPTER 7

# Rails Scaffolding

# RAILS SCAFFOLDING

Using scaffolding you can create the framework for very sophisticated applications in a short space of time.

If you want to create the models, view, and controller for a new application in a single operation, scaffolding is the tool for the job.

You can run several **rails g scaffold** commands to construct each of the elements for the application.

Scaffolding is designed to follow the CRUD and DRY principles, and it forms a good basis for standard rails coding.

But do remember it is someone else’s ‘view’ of what is good and bad.

There is still, however, a fair amount of development work required to effectively join it all together as a complete application, and to have it function exactly as required.

You may need to add validation and error handling for the Model, add associations between models to support your query and display views, customise the views and maybe add extra views to the application.

There is also the issue of ‘appearance’ from a web perspective, Rails supports both standard css and Sass. This is not covered on this course, but it is easy to work with by anyone with previous css experience. The files are located in the **app\assets** directory.

You may also, for example, run a **rails g controller landing index** command to produce the ‘front’ page for the application as a whole; this will generate a controller and a basic index view as a frame work.

The main benefit from using scaffold is the amount of time and effort saved by the developer in creating the basic frame work for the application.

## Quick start

This time we are going to let rails build our guestbook using scaffolding.

As before we have to initialize our application in our \sites directory using:

rails new guest2

cd into the new application directory **\sites\guest2**

Run:

rails g scaffold Person name:string

rake db:migrate

rails server

Got to the URL

**http://localhost:3000/people**

There is your new application, albeit somewhat simple, built with just 4 lines of rails commands.

This means that you can create fairly complex applications very quickly and get them in front of the users ready for further development, such as validation, layout customisation etc.

## Under the ‘hood’

Using our quick start application we will examine exactly what the scaffolding has produced for us.

### Routing

The scaffold created a routes.rb file to support the application. As scaffolding is designed to meet the CRUD approach to web application design, it has simply created a resources entry rather than entries for all the individual elements. From this single entry the necessary routes can be inferred.

The line in the routes.rb file is simply **resources :people**, if we examine the routes using rails routes we get the following:

Prefix Verb URI Pattern Controller#Action

people GET /people(.:format) people#index

POST /people(.:format) people#create

new\_person GET /people/new(.:format) people#new

edit\_person GET /people/:id/edit(.:format) people#edit

person GET /people/:id(.:format) people#show

PATCH /people/:id(.:format) people#update

PUT /people/:id(.:format) people#update

DELETE /people/:id(.:format) people#destroy

We now have the routes needed to Create, Read, Update and Destroy our data.

### Database

The migration file needed for the database and table to be created has been created in db\migrate and contains the following:

class CreatePeople < ActiveRecord::Migration

def change

create\_table :people do |t|

t.string :name

t.timestamps

end

end

end

By simply executing:

rake db:migrate

The underlying database and table will be created.

### Model

A basic outline model file apps\model\person.rb has been created

class Person < ActiveRecord::Base

end

### Controller

The file app\controllers\people\_controller created by scaffold is shown below.

It has all the required entries for each of the routes define in the application.

class PeopleController < ApplicationController

**before\_action :set\_person, only: [:show, :edit, :update, :destroy]**

# GET /people

# GET /people.json

def index

@people = Person.all

end

# GET /people/1

# GET /people/1.json

def show

end

# GET /people/new

def new

@person = Person.new

end

# GET /people/1/edit

def edit

end

# POST /people

# POST /people.json

def create

@person = Person.new(person\_params)

respond\_to do |format|

if @person.save

format.html { redirect\_to @person, notice: 'Person was successfully created.' }

format.json { render :show, status: :created, location: @person }

else

format.html { render :new }

format.json { render json: @person.errors, status: :unprocessable\_entity }

end

end

end

# PATCH/PUT /people/1

# PATCH/PUT /people/1.json

def update

respond\_to do |format|

if @person.update(person\_params)

format.html { redirect\_to @person, notice: 'Person was successfully updated.' }

format.json { render :show, status: :ok, location: @person }

else

format.html { render :edit }

format.json { render json: @person.errors, status: :unprocessable\_entity }

end

end

end

# DELETE /people/1

# DELETE /people/1.json

def destroy

@person.destroy

respond\_to do |format|

format.html { redirect\_to people\_url, notice: 'Person was successfully destroyed.' }

format.json { head :no\_content }

end

end

**private**

**# Use callbacks to share common setup or constraints between actions.**

**def set\_person**

**@person = Person.find(params[:id])**

**end**

**# Never trust parameters from the scary internet, only allow the white list through.**

**def person\_params**

**params.require(:person).permit(:name)**

**end**

end

Rails 6 now has additional code in the template to help with validation and some protection from malicious data, see the bold text above.

### Views

Scaffolding has created the 5 ‘views’ necessary to ‘run’ the application.

The index view, apps\views\people\index.html.erb, is shown below.

<p id="notice"><%= notice %></p>

<h1>People</h1>

<table>

<thead>

<tr>

<th>Name</th>

<th colspan="3"></th>

</tr>

</thead>

<tbody>

<% @people.each do |person| %>

<tr>

<td><%= person.name %></td>

<td><%= link\_to 'Show', person %></td>

<td><%= link\_to 'Edit', edit\_person\_path(person) %></td>

<td><%= link\_to 'Destroy', person, method: :delete, data: { confirm: 'Are you sure?' } %></td>

</tr>

<% end %>

</tbody>

</table>

<br>

<%= link\_to 'New Person', new\_person\_path %>

The scaffold building method supports the 'DRY' approach by using a pre-defined input form that id 'included' as necessary.

The \_form.html.erb file is shown below:

<%= form\_for(person) do |f| %>

<% if person.errors.any? %>

<div id="error\_explanation">

<h2><%= pluralize(person.errors.count, "error") %> prohibited this person from being saved:</h2>

<ul>

<% person.errors.full\_messages.each do |message| %>

<li><%= message %></li>

<% end %>

</ul>

</div>

<% end %>

<div class="field">

<%= f.label :name %>

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

</div>

<div class="actions">

<%= f.submit %>

</div>

<% end %>

As shown before we could add the :required => true parameter to the input text field to avoid blank records being added to the database.

<div class="field">

<%= f.label :name %>

<%= f.text\_field :name, :required => true %>

</div>

## WORKSHOP 2

You have been tasked by your employer, Company X, to produce a small web-based application using Ruby and Rails to allow an Administrator to create and store information about the company departments and employees.

1. Name and location of the department, the locations are either Office or Factory.
2. The Administrator also needs to be able to Add, Update, View and Delete Departments
3. The Administrator needs to be able to Add, Update, View and Delete Employees and assign them to a department
4. View a list of employees in each department.

Notes:

Department record must contain:

**Name**

**Location**

Employee record must contain:

**Name**

**Title**

**Extension Number**

**Department**

An employee can only be a member of one department.

The application needs a front page to allow the administrator to either add/view a new department or employee.

Other application navigation or appearance is at the discretion of the developer.

# Workshop 2 solution

You have been tasked by your employer, Company X, to produce a small web based application using Ruby and Rails to allow an Administrator to create and store information about the company departments and employees.

1. Name and location of the department, the locations are either Office or Factory.
2. The Administrator also needs to be able to Add, Update, View and Delete Departments
3. The Administrator needs to be able to Add, Update, View and Delete Employees and assign them to a department
4. View a list of employees in each department.

Notes:

Department record must contain:

**Department Name**

**Location**

Employee record must contain:

**Name**

**Title**

**Extension Number**

**Department**

An employee can only be a member of one department.

The application needs a front page to allow the administrator to either add/view a new department or employee.

Other application navigation or appearance is at the discretion of the developer.

Assuming project home of **/sites**

Create the rails project:

rails new companyx

cd into companyx

Create the model and controller:

rails g scaffold department name:text location:text

rails g scaffold employee name:text title:text phone:text department:references

rails g controller landing index

Create database and table:

rake db:create

rake db:migrate

Modify the routes.rb and the puma.rb to allow the site to be access as http://localhost/

vi config/routes.rb

replace get entry with:

root ‘landing#index'

Test routes with:

rails routes

## Controllers

### app/controllers/departments\_controller.rb

class DepartmentsController < ApplicationController

before\_action :set\_department, only: [:show, :edit, :update, :destroy]

# GET /departments

# GET /departments.json

def index

@departments = Department.all

end

# GET /departments/1

# GET /departments/1.json

def show

**@employees = Employee.where("department\_id = ?", params[:id])**

end

# GET /departments/new

def new

@department = Department.new

end

# GET /departments/1/edit

def edit

end

# POST /departments

# POST /departments.json

def create

@department = Department.new(department\_params)

respond\_to do |format|

if @department.save

format.html { redirect\_to @department, notice: 'Department was successfully created.' }

format.json { render :show, status: :created, location: @department }

else

format.html { render :new }

format.json { render json: @department.errors, status: :unprocessable\_entity }

end

end

end

# PATCH/PUT /departments/1

# PATCH/PUT /departments/1.json

def update

respond\_to do |format|

if @department.update(department\_params)

format.html { redirect\_to @department, notice: 'Department was successfully updated.' }

format.json { render :show, status: :ok, location: @department }

else

format.html { render :edit }

format.json { render json: @department.errors, status: :unprocessable\_entity }

end

end

end

# DELETE /departments/1

# DELETE /departments/1.json

def destroy

@department.destroy

respond\_to do |format|

format.html { redirect\_to departments\_url, notice: 'Department was successfully destroyed.' }

format.json { head :no\_content }

end

end

private

# Use callbacks to share common setup or constraints between actions.

def set\_department

@department = Department.find(params[:id])

end

# Never trust parameters from the scary internet, only allow the white list through.

def department\_params

params.require(:department).permit(:name, :location)

end

end

### app/controllers/employees\_controller.rb

class EmployeesController < ApplicationController

before\_action :set\_employee, only: [:show, :edit, :update, :destroy]

# GET /employees

# GET /employees.json

def index

@employees = Employee.all

end

# GET /employees/1

# GET /employees/1.json

def show

end

# GET /employees/new

def new

@employee = Employee.new

end

# GET /employees/1/edit

def edit

end

# POST /employees

# POST /employees.json

def create

@employee = Employee.new(employee\_params)

respond\_to do |format|

if @employee.save

format.html { redirect\_to @employee, notice: 'Employee was successfully created.' }

format.json { render :show, status: :created, location: @employee }

else

format.html { render :new }

format.json { render json: @employee.errors, status: :unprocessable\_entity }

end

end

end

# PATCH/PUT /employees/1

# PATCH/PUT /employees/1.json

def update

respond\_to do |format|

if @employee.update(employee\_params)

format.html { redirect\_to @employee, notice: 'Employee was successfully updated.' }

format.json { render :show, status: :ok, location: @employee }

else

format.html { render :edit }

format.json { render json: @employee.errors, status: :unprocessable\_entity }

end

end

end

# DELETE /employees/1

# DELETE /employees/1.json

def destroy

@employee.destroy

respond\_to do |format|

format.html { redirect\_to employees\_url, notice: 'Employee was successfully destroyed.' }

format.json { head :no\_content }

end

end

private

# Use callbacks to share common setup or constraints between actions.

def set\_employee

@employee = Employee.find(params[:id])

end

# Never trust parameters from the scary internet, only allow the white list through.

def employee\_params

params.require(:employee).permit(:name, :phone, :department\_id)

end

end

## Views

As scaffolding creates the basic views only those that require addional code are shown in this solution.

### app/views/departments

show.html.erb

<p id="notice"><%= notice %></p>

<h1>Department</h1>

<p>

<strong>Name:</strong>

<%= @department.name %>

</p>

<p>

<strong>Location:</strong>

<%= @department.location %>

</p>

<h2> Departmental Employees</h2>

<table>

<thead>

<tr>

<th>Name</th>

<th>Phone</th>

<th colspan="3"></th>

</tr>

</thead>

<tbody>

<% @employees.each do |e| %>

<tr>

<td><%= e.name %></td>

<td><%= e.phone %></td>

</tr>

<% end %>

</tbody>

</table>

### app/views/employees

index.html.erb

<p id="notice"><%= notice %></p>

<h1>Employees</h1>

<table>

<thead>

<tr>

<th>Name</th>

<th>Phone</th>

<th>Department</th>

<th colspan="3"></th>

</tr>

</thead>

<tbody>

<% @employees.each do |employee| %>

<tr>

<td><%= employee.name %></td>

<td><%= employee.phone %></td>

<td><%= employee.department.name %></td>

<% end %>

<td><%= link\_to 'Show', employee %></td>

<td><%= link\_to 'Edit', edit\_employee\_path(employee) %></td>

<td><%= link\_to 'Destroy', employee, method: :delete, data: { confirm: 'Are you sure?' } %></td>

</tr>

<% end %>

</tbody>

</table>

<br>

show.html.erb

<p id="notice"><%= notice %></p>

<p>

<strong>Name:</strong>

<%= @employee.name %>

</p>

<p>

<strong>Phone:</strong>

<%= @employee.phone %>

</p>

<p>

<strong>Department:</strong>

<%= @employee.department.name %>

</p>

\_form.html.erb

<%= form\_with(model: employee, local: true) do |form| %>

<% if employee.errors.any? %>

<div id="error\_explanation">

<h2><%= pluralize(employee.errors.count, "error") %> prohibited this employee from being saved:</h2>

<ul>

<% employee.errors.full\_messages.each do |message| %>

<li><%= message %></li>

<% end %>

</ul>

</div>

<% end %>

<div class="field">

<%= form.label :name %>

<%= form.text\_area :name %>

</div>

<div class="field">

<%= form.label :title %>

<%= form.text\_area :title %>

</div>

<div class="field">

<%= form.label :phone %>

<%= form.text\_area :phone %>

</div>

<div class="field">

<%= form.label :department\_id %>

<%= form.collection\_select(:department\_id, Department.all, :id, :name, prompt: true )%>

</div>

<div class="actions">

<%= form.submit %>

</div>

<% end %>

## Global assets and Styles

### app/views/layouts/application.html.erb

<!DOCTYPE html>

<html>

<head>

<title>Company</title>

<%= csrf\_meta\_tags %>

<%= stylesheet\_link\_tag 'application', media: 'all', 'data-turbolinks-track': 'reload' %>

<%= javascript\_include\_tag 'application', 'data-turbolinks-track': 'reload' %>

</head>

<body>

<%= yield %>

</body>

<div class="footer">

<footer>

<ul>

<li><%= link\_to 'Show all Employees', employees\_path %></li>

<li><%= link\_to 'Create new Employee', new\_employee\_path %></li>

<li><%= link\_to 'Show all Departments', departments\_path %></li>

<li><%= link\_to 'Create new Department', new\_department\_path%></li>

<li><%= link\_to 'home', root\_path%></li>

</ul>

</footer>

</div>

</html>

### app/assets/stylesheets/application.css

/\*

\* This is a manifest file that'll be compiled into application.css, which will include all the files

\* listed below.

\*

\* Any CSS and SCSS file within this directory, lib/assets/stylesheets, vendor/assets/stylesheets,

\* or any plugin's vendor/assets/stylesheets directory can be referenced here using a relative path.

\*

\* You're free to add application-wide styles to this file and they'll appear at the bottom of the

\* compiled file so the styles you add here take precedence over styles defined in any other CSS/SCSS

\* files in this directory. Styles in this file should be added after the last require\_\* statement.

\* It is generally better to create a new file per style scope.

\*

\*= require\_tree .

\*= require\_self

\*/

.footer {

position: fixed;

left: 0;

bottom: 0;

width: 100%;

background-color: PowderBlue;

color: white;

text-align: center;

}

body {

background-color: #88918d;

}

H1 {

color: #ffff66

}