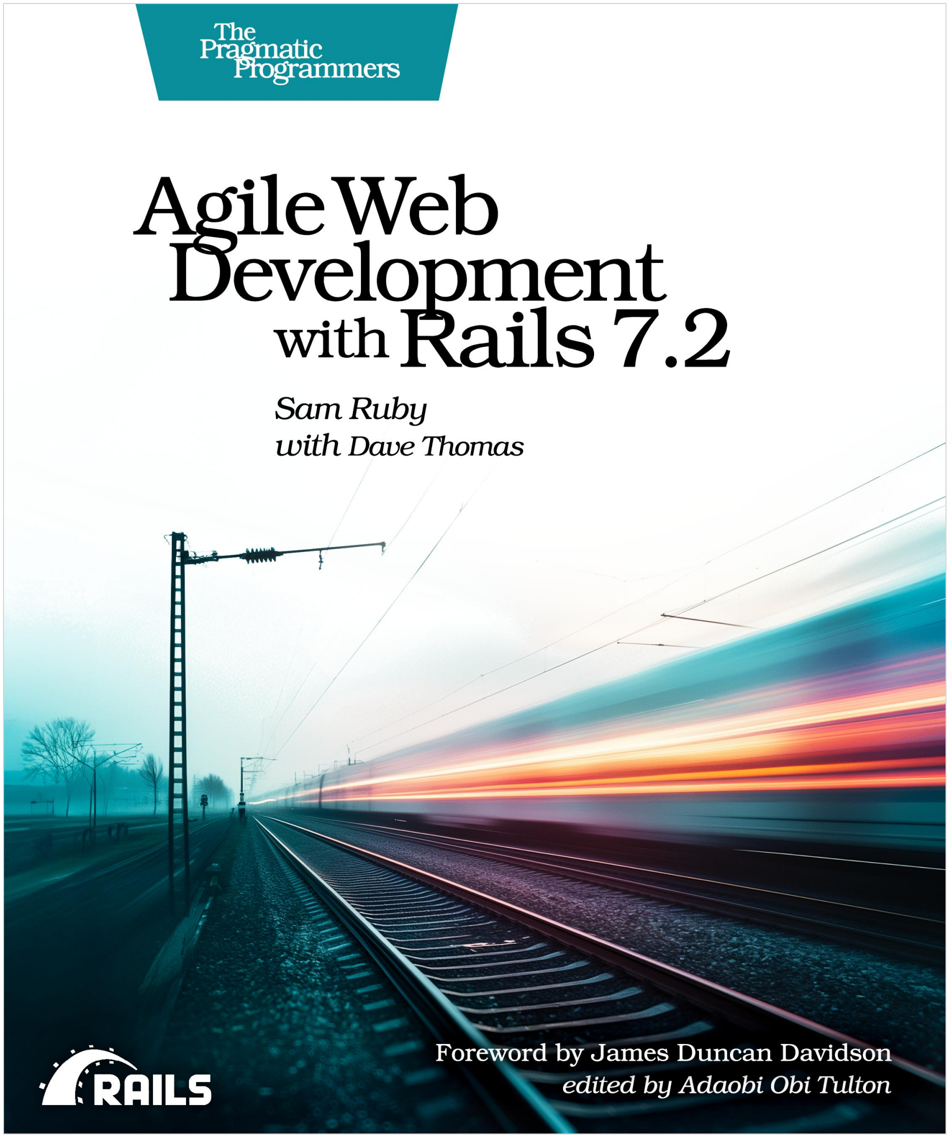


The Pragmatic Programmers

Agile Web Development with Rails 7.2

Sam Ruby
with Dave Thomas

Foreword by James Duncan Davidson
edited by Adaobi Obi Tulton

Iteration J1: Selecting the Locale

We start by creating a new configuration file that encapsulates our knowledge of what locales are available and which one is to be used as the default:

rails72/depot_t/config/initializers/i18n.rb

```
#encoding: utf-8
I18n.default_locale = :en

LANGUAGES = [
  [ "English",                  "en" ],
  [ "Espa&ntilde;ol".html_safe, "es" ]
]
```

This code is doing two things.

The first thing it does is use the `I18n` module to set the default locale. `I18n` is a funny name, but it sure beats typing out *internationalization* all the time. Internationalization, after all, starts with an *i*, ends with an *n*, and has eighteen letters in between.

Then the code defines a list of associations between display names and locale names. Unfortunately, all we have available at the moment is a U.S. keyboard, and Español has a character that can't be directly entered via our keyboard. Different operating systems have different ways of dealing with this, and often the easiest way is to copy and paste the correct text from a website. If you do this, make sure your editor is configured for UTF-8. Meanwhile, we've opted to use the HTML equivalent of the *n con tilde* character in Spanish. If we didn't do anything else, the markup itself would be shown. But by calling `html_safe`, we inform Rails that the string is safe to be interpreted as containing HTML.

For Rails to pick up this configuration change, the server needs to be restarted.

Since each page that's translated will have an `en` and an `es` version (for now —more will be added later), it makes sense to include this in the URL. Let's plan to put the locale up front, make it optional, and have it default to the current locale, which in turn will default to English.

To implement this cunning plan, let's start by modifying `config/routes.rb`:

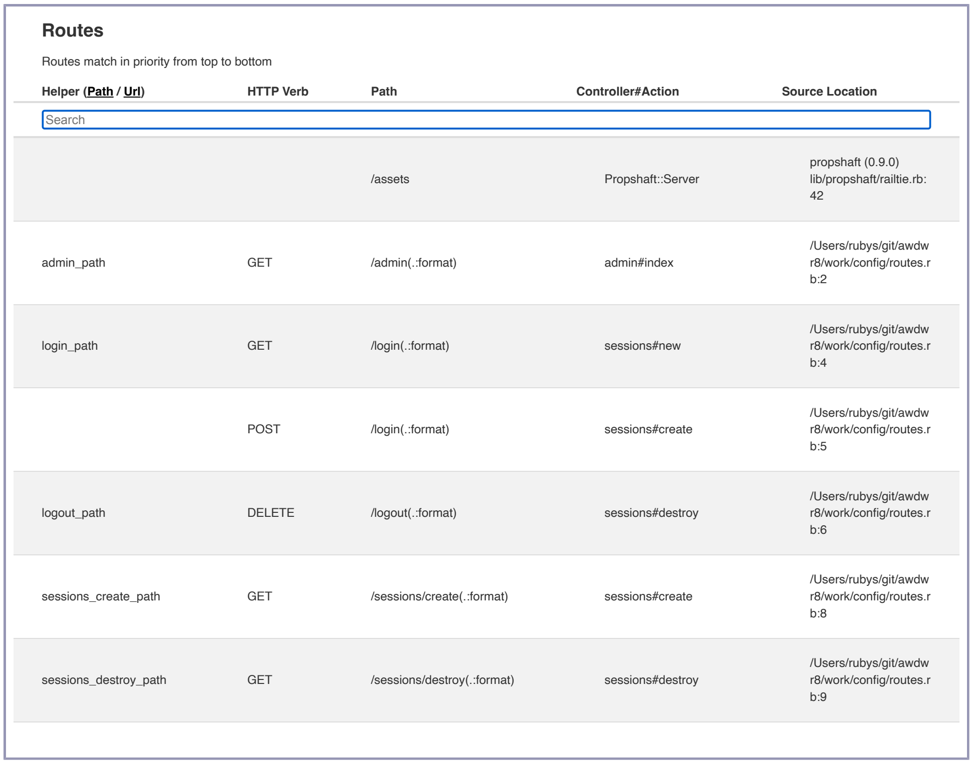
rails72/depot_t/config/routes.rb

```
  Rails.application.routes.draw do
    get "admin" => "admin#index"
    controller :sessions do
      get  "login" => :new
      post "login" => :create
      delete "logout" => :destroy
    end
    get "sessions/create"
    get "sessions/destroy"
    get "up" => "rails/health#show", as: :rails_health_check
    get "service-worker" => "rails/pwa#service_worker", as: :pwa_service_worker
    get "manifest" => "rails/pwa#manifest", as: :pwa_manifest

    resources :users
    resources :products do
      get :who_bought, on: :member
    end

➤   scope "(:locale)" do
      resources :orders
      resources :line_items
      resources :carts
➤     root "store#index", as: "store_index", via: :all
➤   end
  end
```

We've nested our resources and root declarations inside a scope declaration for :locale. Furthermore, :locale is in parentheses, which is the way to say that it's optional. Note that we didn't choose to put the administrative and session functions inside this scope, because it's not our intent to translate them at this time.

What this means is that http://localhost:3000/ will use the default locale (namely, English) and therefore be routed exactly the same as http://localhost:3000/en. http://localhost:3000/es will route to the same controller and action, but we'll want this to cause the locale to be set differently.

At this point, we've made a lot of changes to config.routes, and with the nesting and all the optional parts to the path, the gestalt might be hard to visualize. Never fear—when running a server in development mode, Rails provides a visual aid. All you need to do is navigate to http://localhost:3000/rails/info/routes, and you'll see a list of all your routes. You can even filter the list, as shown in the screenshot on page 5, to quickly find the route you're interested in. More information on the fields shown in this table can be found in the description of rake routes on page ?.



Routes

Routes match in priority from top to bottom

Helper (Path / Url)	HTTP Verb	Path	Controller#Action	Source Location
Search				
		/assets	Propshaft::Server	propshaft (0.9.0) lib/propshaft/railtie.rb:42
admin_path	GET	/admin(.:format)	admin#index	/Users/rubys/git/awdwr8/work/config/routes.rb:2
login_path	GET	/login(.:format)	sessions#new	/Users/rubys/git/awdwr8/work/config/routes.rb:4
	POST	/login(.:format)	sessions#create	/Users/rubys/git/awdwr8/work/config/routes.rb:5
logout_path	DELETE	/logout(.:format)	sessions#destroy	/Users/rubys/git/awdwr8/work/config/routes.rb:6
sessions_create_path	GET	/sessions/create(.:format)	sessions#create	/Users/rubys/git/awdwr8/work/config/routes.rb:8
sessions_destroy_path	GET	/sessions/destroy(.:format)	sessions#destroy	/Users/rubys/git/awdwr8/work/config/routes.rb:9

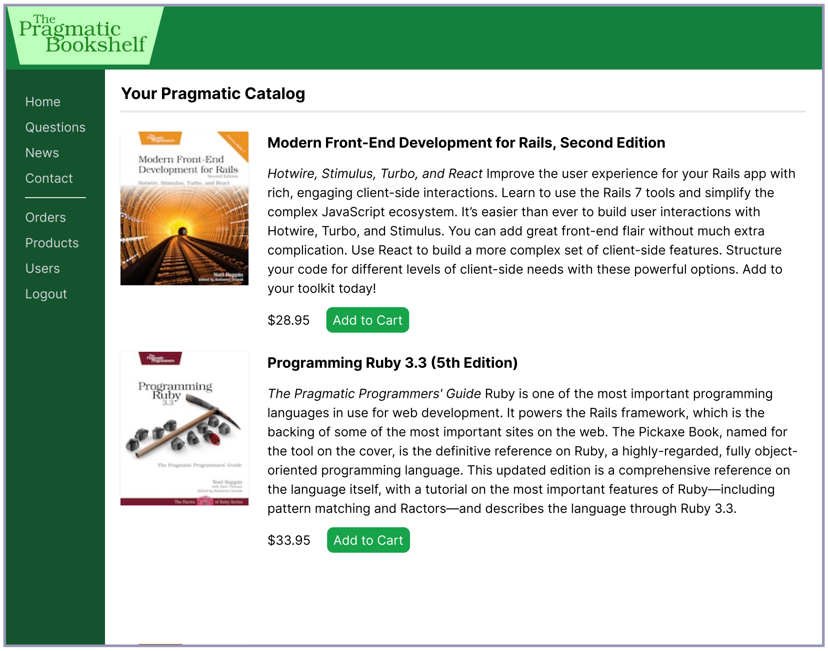
With the routing in place, we're ready to extract the locale from the parameters and make it available to the application. To do this, we need to create a before_action callback. The logical place to do this is in the common base class for all of our controllers, which is ApplicationController:

rails72/depot_t/app/controllers/application_controller.rb

```
  class ApplicationController < ActionController::Base
➤   before_action :set_i18n_locale_from_params
    # ...
    protected
➤     def set_i18n_locale_from_params
➤       if params[:locale]
➤         if I18n.available_locales.map(&:to_s).include?(params[:locale])
➤           I18n.locale = params[:locale]
➤         else
➤           flash.now[:notice] =
➤             "#{params[:locale]} translation not available"
➤           logger.error flash.now[:notice]
➤         end
➤       end
➤     end
  end
```

This set_i18n_locale_from_params does pretty much what it says: it sets the locale from the params, but only if there's a locale in the params; otherwise, it leaves the current locale alone. Care is taken to provide a message for both the user and the administrator when a failure occurs.

With this in place, we can see the results in the following screenshot of navigating to http://localhost:3000/en.



Your Pragmatic Catalog

Modern Front-End Development for Rails, Second Edition

Hotwire, Stimulus, Turbo, and React Improve the user experience for your Rails app with rich, engaging client-side interactions. Learn to use the Rails 7 tools and simplify the complex JavaScript ecosystem. It's easier than ever to build user interactions with Hotwire, Turbo, and Stimulus. You can add great front-end flair without much extra complication. Use React to build a more complex set of client-side features. Structure your code for different levels of client-side needs with these powerful options. Add to your toolkit today!

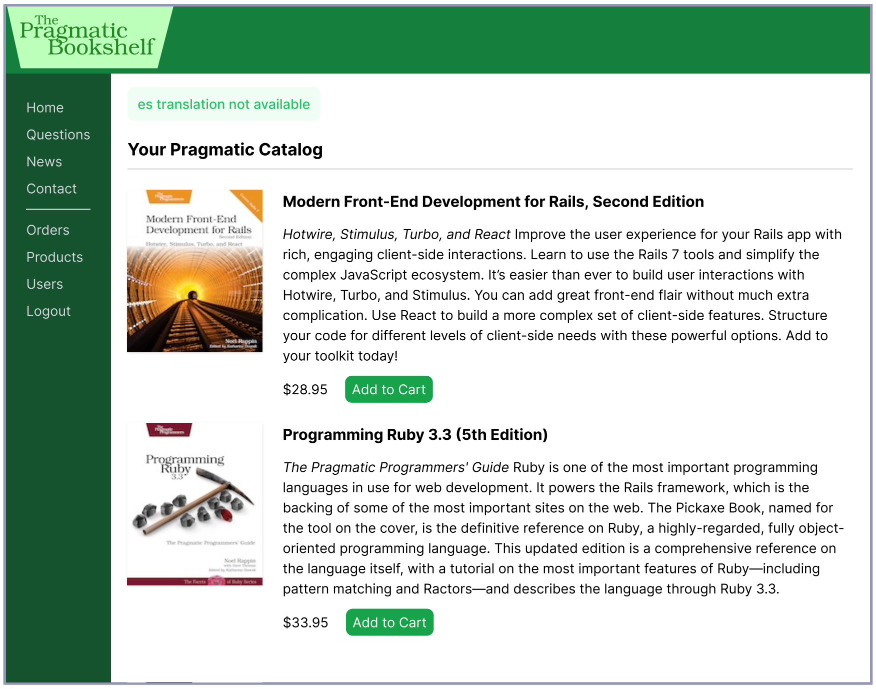
$28.95 Add to Cart

Programming Ruby 3.3 (5th Edition)

The Pragmatic Programmers' Guide Ruby is one of the most important programming languages in use for web development. It powers the Rails framework, which is the backing of some of the most important sites on the web. The Pickaxe Book, named for the tool on the cover, is the definitive reference on Ruby, a highly-regarded, fully object-oriented programming language. This updated edition is a comprehensive reference on the language itself, with a tutorial on the most important features of Ruby—including pattern matching and Ractors—and describes the language through Ruby 3.3.

$33.95 Add to Cart

At this point, the English version of the page is available both at the root of the website and at pages that start with /en. If you try another language code, say "es" (or Spanish), you can see that an error message appears saying no translations are available. The screenshot on page 7 shows what this might look like when navigating to http://localhost:3000/es:



es translation not available

Your Pragmatic Catalog

Modern Front-End Development for Rails, Second Edition

Hotwire, Stimulus, Turbo, and React Improve the user experience for your Rails app with rich, engaging client-side interactions. Learn to use the Rails 7 tools and simplify the complex JavaScript ecosystem. It's easier than ever to build user interactions with Hotwire, Turbo, and Stimulus. You can add great front-end flair without much extra complication. Use React to build a more complex set of client-side features. Structure your code for different levels of client-side needs with these powerful options. Add to your toolkit today!

$28.95 Add to Cart

Programming Ruby 3.3 (5th Edition)

The Pragmatic Programmers' Guide Ruby is one of the most important programming languages in use for web development. It powers the Rails framework, which is the backing of some of the most important sites on the web. The Pickaxe Book, named for the tool on the cover, is the definitive reference on Ruby, a highly-regarded, fully object-oriented programming language. This updated edition is a comprehensive reference on the language itself, with a tutorial on the most important features of Ruby—including pattern matching and Ractors—and describes the language through Ruby 3.3.

$33.95 Add to Cart