

Extracted from:

Agile Web Development with Rails 7

This PDF file contains pages extracted from *Agile Web Development with Rails 7*, published by the Pragmatic Bookshelf. For more information or to purchase a paperback or PDF copy, please visit <http://www.pragprog.com>.

Note: This extract contains some colored text (particularly in code listing). This is available only in online versions of the books. The printed versions are black and white. Pagination might vary between the online and printed versions; the content is otherwise identical.

Copyright © 2023 The Pragmatic Programmers, LLC.

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

The Pragmatic Bookshelf

Raleigh, North Carolina

The
Pragmatic
Programmers

Agile Web Development with Rails 7

*Sam Ruby
with Dave Thomas*



Foreword by
James Duncan Davidson

Edited by Adaobi Obi Tulton

Agile Web Development with Rails 7

Sam Ruby

The Pragmatic Bookshelf

Raleigh, North Carolina



Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and The Pragmatic Programmers, LLC was aware of a trademark claim, the designations have been printed in initial capital letters or in all capitals. The Pragmatic Starter Kit, The Pragmatic Programmer, Pragmatic Programming, Pragmatic Bookshelf, PragProg and the linking *g* device are trademarks of The Pragmatic Programmers, LLC.

Every precaution was taken in the preparation of this book. However, the publisher assumes no responsibility for errors or omissions, or for damages that may result from the use of information (including program listings) contained herein.

For our complete catalog of hands-on, practical, and Pragmatic content for software developers, please visit <https://pragprog.com>.

The team that produced this book includes:

CEO: Dave Rankin

COO: Janet Furlow

Managing Editor: Tammy Coron

Development Editor: Adaobi Obi Tulton

Copy Editor: L. Sakhi MacMillan

Indexing: Potomac Indexing, LLC

Layout: Gilson Graphics

Founders: Andy Hunt and Dave Thomas

For sales, volume licensing, and support, please contact support@pragprog.com.

For international rights, please contact rights@pragprog.com.

Copyright © 2023 The Pragmatic Programmers, LLC.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

ISBN-13: 978-1-68050-929-8

Encoded using the finest acid-free high-entropy binary digits.

Book version: P1.0—May 2023

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:

```
rails7/depot_t/config/initializers/i18n.rb
#encoding: utf-8
I18n.default_locale = :en

LANGUAGES = [
  ['English', 'en'],
  ["Español", :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`:

```
rails7/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'
```

```
  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 7](#), 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	HTTP Verb	Path	Controller#Action
Path / Url		Path Match	
admin_path	GET	/admin(.:format)	admin#index
login_path	GET	/login(.:format)	sessions#new
	POST	/login(.:format)	sessions#create
logout_path	DELETE	/logout(.:format)	sessions#destroy
sessions_create_path	GET	/sessions/create(.:format)	sessions#create
sessions_destroy_path	GET	/sessions/destroy(.:format)	sessions#destroy
users_path	GET	/users(.:format)	users#index
	POST	/users(.:format)	users#create
new_user_path	GET	/users/new(.:format)	users#new
edit_user_path	GET	/users/:id/edit(.:format)	users#edit
user_path	GET	/users/:id(.:format)	users#show
	PATCH	/users/:id(.:format)	users#update
	PUT	/users/:id(.:format)	users#update
	DELETE	/users/:id(.:format)	users#destroy
who_bought_product_path	GET	/products/:id/who_bought(.:format)	products#who_bought
products_path	GET	/products(.:format)	products#index
	POST	/products(.:format)	products#create
new_product_path	GET	/products/new(.:format)	products#new
edit_product_path	GET	/products/:id/edit(.:format)	products#edit

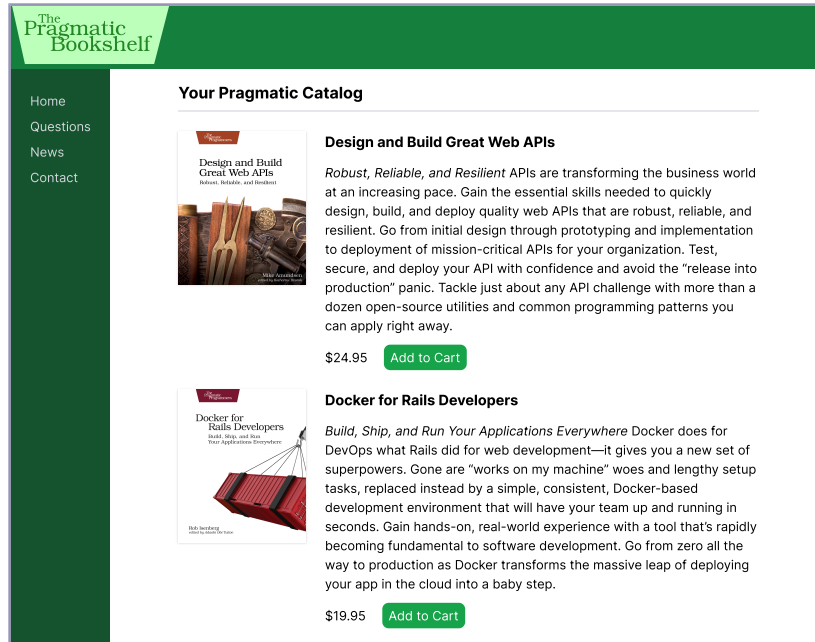
With the routing in place, we're ready to extract the locale from the parameters and make it available to the application. To do this, we need to create a `before_action` callback. The logical place to do this is in the common base class for all of our controllers, which is `ApplicationController`:

```
rails7/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`.



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 9](#) shows what this might look like when navigating to `http://localhost:3000/es`:

The Pragmatic Bookshelf

Home


Questions

News

Contact

es translation not available

Your Pragmatic Catalog



Design and Build Great Web APIs


Robust, Reliable, and Resilient

Design and Build Great Web APIs

Robust, Reliable, and Resilient APIs are transforming the business world at an increasing pace. Gain the essential skills needed to quickly design, build, and deploy quality web APIs that are robust, reliable, and resilient. Go from initial design through prototyping and implementation to deployment of mission-critical APIs for your organization. Test, secure, and deploy your API with confidence and avoid the “release into production” panic. Tackle just about any API challenge with more than a dozen open-source utilities and common programming patterns you can apply right away.

\$24.95

Add to Cart



Docker for Rails Developers

Build, Ship, and Run Your Applications Everywhere

Docker for Rails Developers

Build, Ship, and Run Your Applications Everywhere Docker does for DevOps what Rails did for web development—it gives you a new set of superpowers. Gone are “works on my machine” woes and lengthy setup tasks, replaced instead by a simple, consistent, Docker-based development environment that will have your team up and running in seconds. Gain hands-on, real-world experience with a tool that’s rapidly becoming fundamental to software development. Go from zero all the way to production as Docker transforms the massive leap of deploying your app in the cloud into a baby step.

• Click [HERE](#) to purchase this book now. discuss