once-campfire のコードリーディングのすゝめ

maimu(@maimux2x)

2026.02.19/RailsTokyo#3

maimu(@maimux2x)
永和システムマネジメント所属
好きな Rails メソッドは index_by


しんめ.rb

今日お伝えしたいこと

once-campfire の
コードリーディングのすゝめ

勉強になった・印象的だった箇所を
中心にお伝えします！

準備編

once-campfire とは

Campfire 1.4.3

Super simple group chat, without a subscription. And you get the code, too.

If you've used Slack or Teams, you already know how to chat with Campfire. Download, run one command, and host your whole company on your own server. It's yours.

https://once.com/campfire

- サブスクリプション不要の超シンプルなグループチャット
- ソースコードが公開されている

once-campfire の特徴

the basics done right

基本的な機能がしっかり揃っている
ルーム作成
DM
@mentions
検索
モバイル対応

Self-hosted / Own your data

自分たちでホストし、一度手に入れたら使い続けることができる
データを自分で所有できる

Anti-subscription / Anti-complexity

チャットツールに永遠に課金するのか？
機能が複雑すぎないか？

As educational material

学習可能な高品質コードとして公開

Getting Real

The smarter, faster, easier way to build a successful web application

Continue from where you left off →

---

https://basecamp.com/gettingreal

読んでいく方向性を考える

サービスの中心となるリソースを押さえる

このサービスは「誰／何」を中心に設計されているか

「誰」が動作の主体か？

```ruby
# config/routes.rb
resources :users, only: :show do
  scope module: "users" do
    resource :avatar, only: %i[ show destroy ]
    resource :ban, only: %i[ create destroy ]
    .
    .
    .
  end
end
```

users が定義されている

「何」を扱うサービスか？

```ruby
# config/routes.rb

resources :rooms do
  resources :messages

  …
  scope module: "rooms" do
    .
    .
    .
  end

  get "@:message_id", to: "rooms#show", as: :at_message
end
```

チャットアプリという点からも rooms と messages が「何」の中心と読み取れる

Room を中心に User が Message をやり取りする

User / Room / Messages を中心に読み進める

学びになった実装箇所

1. User の表現

https://speakerdeck.com/moro/identifying-user-idenity

```
 # db/schema.rb

create_table "users", force: :cascade do |t|
   t.text "bio"
   t.string "bot_token"

   t.datetime "created_at", null: false
   t.string "email_address"
   t.string "name", null: false    t.string "password_digest"
   t.integer "role", default: 0, null: false
   t.integer "status", default: 0, null: false
   t.datetime "updated_at", null: false
   t.index ["bot_token"], name: "index_users_on_bot_token", unique: true
   t.index ["email_address"], name: "index_users_on_email_address", unique: true
end
```

users テーブルで完結
サブリソースはなさそう

```ruby
# app/models/room.rb

class Room < ApplicationRecord

  has_many :users, through: :memberships
  has_many :messages, dependent: :destroy

  belongs_to :creator, class_name: "User", default: -> { Current.user }

end
```

このサービスでは User は users と creator で表現が2通りある

2. User の認証

```
source "https://rubygems.org"

# Drivers
gem "sqlite3"
gem "redis", "~> 5.4"

# Deployment
gem "puma", "~> 6.6"

# Jobs
gem "resque", "~> 2.7.0"
gem "resque-pool", "~> 0.7.1"
.
.
.
```

認証系の gem は使われてなさそう

```ruby
# config/routes.rb
resource :session do
  scope module: "sessions" do
    resource :transfers, only: %i[ show update ]
  end
end
```

認証に関係しそうなリソースから追う

```ruby
# app/controllers/sessions_controller.rb
class SessionsController < ApplicationController
  allow_unauthenticated_access only: %i[ new create ]
  rate_limit to: 10, within: 3.minutes, only: :create, with: -> {
    render_rejection :too_many_requests
  }

  def new
  end

  def create
    if user = User.active.authenticate_by(
      email_address: params[:email_address],
      password: params[:password]
    )
      start_new_session_for user
```

既視感があるコード・・・

$ bin/rails generate authentication

rails g authenticationから学ぶRails8.0時代の認証

Shinichi Maeshima(@willnet)
Kaigi on Rails 2025

2025/09/27

https://speakerdeck.com/willnet/rails-g-authenticationkaraxue-hurails8-dot-0shi-dai-noren-zheng

Rails 8 の認証機能と完全に同じではない

```ruby
#!/usr/bin/env ruby
require File.expand_path("../../config/environment", File.dirname(__FILE__))

abort "Usage: #{$0} <email-address> <password>" unless ARGV.length == 2

email_address, password = ARGV
if user = User.find_by(email_address: email_address)
  user.update!(password: password)
  puts "Password has been reset"
else
  puts "User not found"
  exit -1
end
```

パスワードを忘れてしまった場合は、管理者がサーバー上でCLIスクリプトを実行する方式

メール送信によるパスワードリセットは実装されていない

app/mailers も存在しない

シンプルさを優先

campfire の思想が見えてきておもしろくなってきました！

3. 関連の拡張

```ruby
# app/models/room.rb
class Room < ApplicationRecord
  has_many :memberships, dependent: :delete_all do
    def grant_to(users)
      room = proxy_association.owner
      Membership.insert_all(Array(users).collect { |user|
      { room_id: room.id, user_id: user.id, involvement: room.default_involvement }
      })
    end

    def revoke_from(users)
      destroy_by user: users
    end

    def revise(granted: [], revoked: [])
      transaction do
        grant_to(granted) if granted.present?
        revoke_from(revoked) if revoked.present?
      end
    end
  end
```

8.5 関連付けの拡張

Railsは、関連付けのプロキシオブジェクト（関連付けを管理する）を拡張するための機能を提供しており、新しいファインダーやクリエーターなどのメソッドを無名モジュール（anonymous module）を介して追加します。この機能により、アプリケーションの特定のニーズに合わせて関連付けをカスタマイズできます。

以下のように、モデル定義内で直接カスタムメソッドを利用することで `has_many` 関連付けを拡張できます。

```
class Author < ApplicationRecord
  has_many :books do
    def find_by_book_prefix(book_number)
      find_by(category_id: book_number[0..2])
    end
  end
end
```

https://railsguides.jp/association_basics.html#%E9%96%A2%E9%80%A3%E4%BB%98%E3%81%91%E3%81%AE%E6%8B%A1%E5%BC%B5

```ruby
room = Room.find(1)

room.memberships.grant_to(users)      # ユーザーにアクセス権を付与
room.memberships.revoke_from(users)   # アクセス権を剥奪
room.memberships.revise(granted: [...], revoked: [...])  # 一括更新
```

インスタンスメソッドで書き換えてみる

```ruby
def grant_memberships_to(users)
  Membership.insert_all(
    Array(users).collect do |user|
      { room_id: id, user_id: user.id, involvement: default_involvement }
    end
  )
end

room = Room.find(1)
room.grant_memberships_to(users)
```

```
# 関連の拡張の場合
room.memberships.grant_to(users)

# インスタンスメソッドの場合
room.grant_memberships_to(users)
```

関連の拡張を使うことで、Room が Membership を介して User の権限を設定する意図が表現しやすそう

once-campfire ならでは？

印象的だった実装箇所

1. bin/setup

```
# Install dependencies
if command -v brew &>/dev/null; then
  step "Installing packages" brew install sqlite ffmpeg mise
elif command -v pacman &>/dev/null; then
  step "Installing packages" sudo pacman -S --noconfirm --needed sqlite ffmpeg mise
elif command -v apt &>/dev/null; then
  step "Installing packages" sudo apt-get install --no-install-recommends -y libsqlite3-0 ffmpeg
fi
```

Arch Linux が想定されているのが珍しい
DHH が Omarchy の開発をしているからかなと思ったり

2. SQLite の全文検索

db/schema.rb を確認しようとしたら...

2025年11月時点

db/structure.sql があった

なぜ？

※今は db/schema.rb に変更されている

6.2.2 `:sql` スキーマダンプを利用する場合

しかし、`db/schema.rb` では、「トリガー」「シーケンス」「ストアドプロシージャ」「チェック制約」などのデータベース固有の項目までは表現できません。

マイグレーションで `execute` を用いれば、RubyマイグレーションDSLでサポートされないデータベース構造も作成できますが、そうしたステートメントはスキーマダンプで再構成されないので、注意が必要です。

https://railsguides.jp/active_record_migrations.html#sql%E3%82%B9%E3%82%AD%E3%83%BC%E3%83%9E%E3%83%80%E3%83%B3%E3%83%97%E3%82%92%E5%88%A9%E7%94%A8%E3%81%99%E3%82%8B%E5%A0%B4%E5%90%88

```
# db/migrate/20231215043540_create_initial_schema.rb

execute <<-SQL
  create virtual table message_search_index using fts5(body, tokenize=porter);
SQL
```

execute で grep してみる

FTS5 とは

FTS5 はSQLite の拡張モジュールでテキストの全文検索を高速に行うための仮想テーブルを提供する

過去の schema.rb では仮想テーブルを扱えなかったため structure.sql を使っていた

Add support for SQLite3 full-text-search and other virtual tables #52354

Merged rafaelfranca merged 1 commit into rails:main from zachasme:sqlite-virtual-tables on Aug 27, 2024

Conversation 23 | Commits 1 | Checks 0 | Files changed 13

zachasme commented on Jul 18, 2024 • edited — Contributor

Motivation / Background

As discussed on rubyonrails-core with **@flavorjones**.

SQLite is a great default database, and even provides a full-text-search module, `fts5`. However, `fts5` is a virtual table module which messes up `schema.rb`. For example, if you create virtual table `searchables` in a migration:

```
class CreateSearchablesTable < ActiveRecord::Migration[8.0]
  def up
    execute 'CREATE VIRTUAL TABLE searchables USING fts5(content)'
  end
end
```

and run `rails db:migrate`, then 6 tables are created, 1 `virtual` and 5 "`shadow`" tables. This is what the result of `SELECT * FROM pragma_table_list ORDER BY name` (relevant part) looks like:

Reviewers
- rafaelfranca
- matthewd

+3 more reviewers
- kevinmcconnell
- willianveiga
- flavorjones

Assignees
No one assigned

Labels
activerecord railties

https://github.com/rails/rails/pull/52354

Modernize scripts #117

Preview | Code

Merged monorkin merged 4 commits into main from modernize-scripts on Dec 1, 2025

Conversation 0 | Commits 4 | Checks 14 | Files changed 5 | +287 -222

Commit 3367ffa | Comments 0 | Submit comments

Filter files...

- config
  - application.rb
- db
  - schema.rb

Switch to using schema.rb

Previously we had to use structure.sql since schema.rb didn't have support for virtual tables that we needed for search. Since Campfire's release virtuals tables have been added to Rails, so there is no need to use structure.sql anymore

Prev | Next | Browse files

monorkin committed on Dec 1, 2025 commit 3367ffa

config/application.rb -3

```
@@ -16,8 +16,5 @@ class Application < Rails::Application
16 16
17 17        # Fallback to English if translation key is missing
18 18        config.i18n.fallbacks = true
19    -
20    -     # Use SQL schema format to include search-related objects
21    -     config.active_record.schema_format = :sql
22 19      end
23 20    end
```

https://github.com/basecamp/once-campfire/pull/117

SQLite FTS5 Extension

▶ Table Of Contents

1. Overview of FTS5

FTS5 is an SQLite virtual table module that provides full-text search functionality to database applications. In their most elementary form, full-text search engines allow the user to efficiently search a large collection of documents for the subset that contain one or more instances of a search term. The search functionality provided to world wide web users by Google is, among other things, a full-text search engine, as it allows users to search for all documents on the web that contain, for example, the term "fts5".

To use FTS5, the user creates an FTS5 virtual table with one or more columns. For example:

```
CREATE VIRTUAL TABLE email USING fts5(sender, title, body);
```

It is an error to add types, constraints or PRIMARY KEY declarations to a CREATE VIRTUAL TABLE statement used to create an FTS5 table. Once created, an FTS5 table may be populated using INSERT, UPDATE or DELETE statements like any other table. Like any other table with no PRIMARY KEY declaration, an FTS5 table has an implicit INTEGER PRIMARY KEY field named rowid.

https://www.sqlite.org/fts5.html

```ruby
# app/models/message/serchable.rb

module Message::Searchable
  extend ActiveSupport::Concern

  included do
    after_create_commit  :create_in_index
    after_update_commit  :update_in_index
    after_destroy_commit :remove_from_index

    scope :search, ->(query) {
      joins(
        "join message_search_index idx on messages.id = idx.rowid"
      ).where("idx.body match ?", query).ordered
    }
  end
```

3. ページネーション

```
# Gemfile

# Other
gem "bcrypt"
gem "web-push"
gem "rqrcode"
gem "rails_autolink"
gem "geared_pagination"
gem "jbuilder"
gem "net-http-persistent"
gem "kredis"
gem "platform_agent"
gem "thruster"
```

ページサイズを固定しないページネーション

1ページ目 → 15件
2ページ目 → 30件
3ページ目 → 50件
4ページ目以降 → 100件

Geared Pagination

Most pagination schemes use a fixed page size. Page 1 returns as many elements as page 2. But that's frequently not the most sensible way to page through a large recordset when you care about serving the initial request as quickly as possible. This is particularly the case when using the pagination scheme in combination with an infinite scrolling UI.

https://github.com/basecamp/geared_pagination

チャットルームで使われていると思いきや

set_page_and_extract_portion_from で grep してみると room では使われてない

```
app/controllers/accounts_controller.rb|8 col 5| set_page_and_extract_portion_from
app/controllers/autocompletable/users_controller.rb|3 col 5| set_page_and_extract_portion_from
app/controllers/accounts/users_controller.rb|5 col 5| set_page_and_extract_portion_from
```

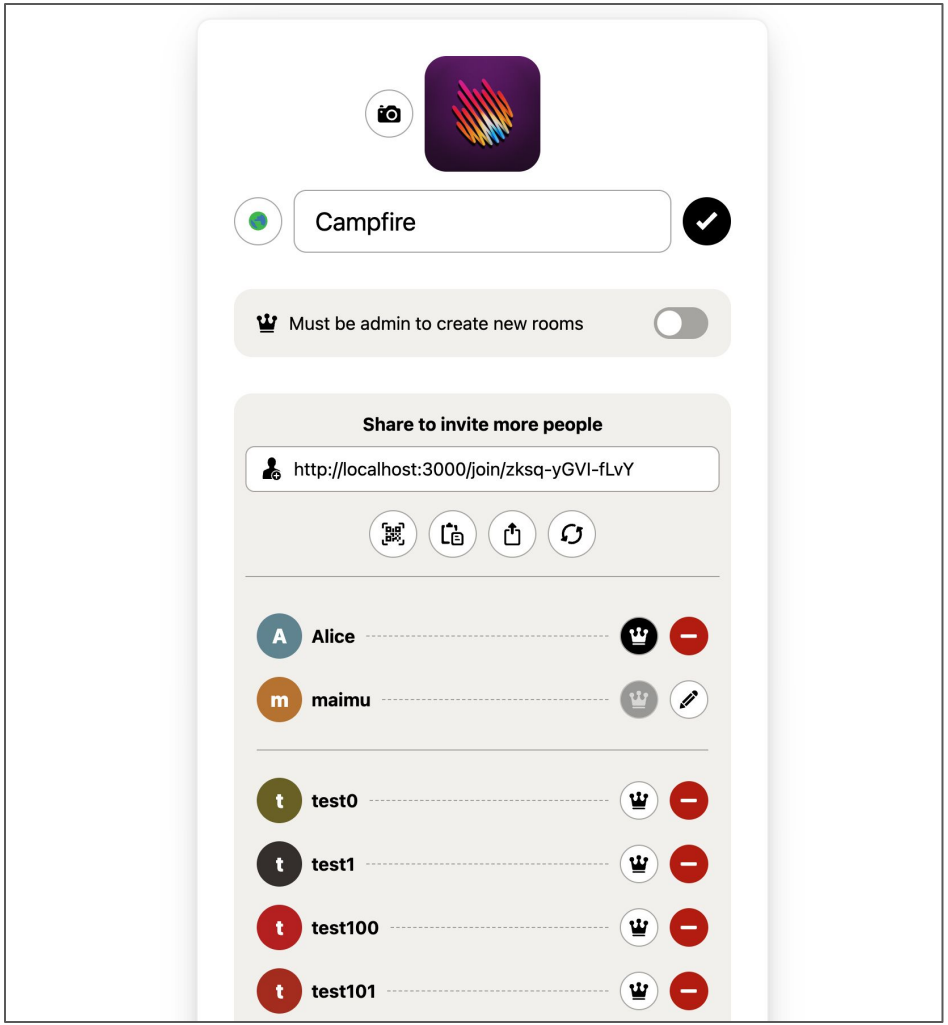
メッセージをスクロールする時の
ページネーションはどうなっている？

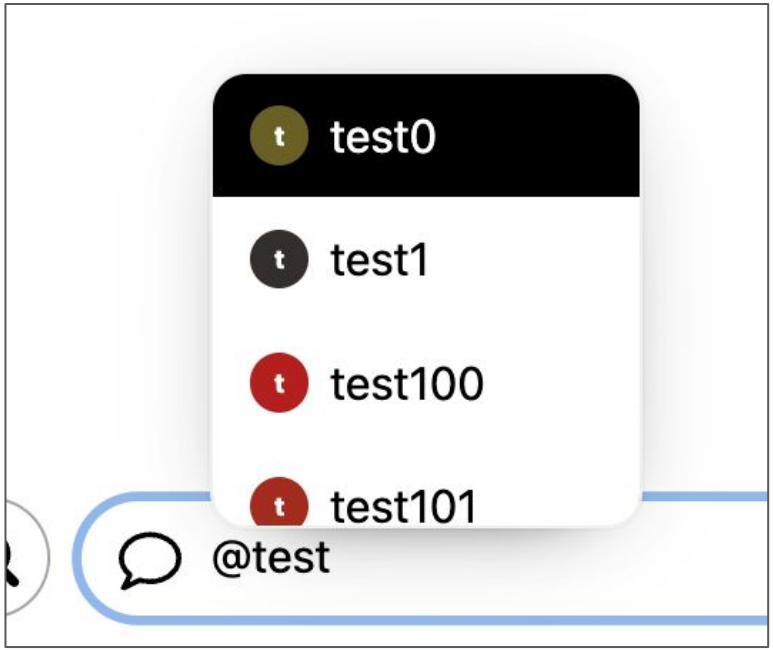
```
#app/controllers/rooms_controller.rb

class RoomsController < ApplicationController
  …
  private
    def find_messages
      messages =
       @room.messages.with_creator.with_attachment_details.with_boosts

      if show_first_message = messages.find_by(id: params[:message_id])
        @messages = messages.page_around(show_first_message)
      else
        @messages = messages.last_page
      end
    end
end
```

message_id が指定されている時はその前後、未指定の場合は最新のメッセージを基準に表示するようになっている

```ruby
# app/models/message/pagination.rb
module Message::Pagination
  extend ActiveSupport::Concern

  PAGE_SIZE = 40

  included do
    …
    scope :before, ->(message) { where("created_at < ?", message.created_at) }
    scope :after, ->(message) { where("created_at > ?", message.created_at) }
    …
  class_methods do
    def page_around(message)
      page_before(message) + [ message ] + page_after(message)
    end

    def paged?
      count > PAGE_SIZE
    end
  end
```

チャットメッセージはパーマリンク対応やカーソルベースの双方向スクロール、リアルタイム更新との連携など、チャット固有の要件があるため独自実装されている





geared_pagination は Accounts に所属する User リストやチャットのメンション時の User の候補リストに部分的に使われている。

最後に

- 文法・設計・思想など様々な観点で読むことができて面白い
- なんでこうなっているんだろう？と深掘りしていくとプログラミングの世界が広がる

今日の発表が once-campfire の
コードリーディングをする
きっかけの一つになれたら嬉しいです