Learn Ruby on Rails Best Practices with One Exercise 🍝

Thiago Araujo

hexdevs.com

You're all here because you want to become even greater Ruby developers.

You want to write professional, squeaky-clean code.

Sometimes you wonder:

- How can I organize a large Ruby on Rails project?
- Where should this piece of business logic live?
- How do I improve this codebase?
- How can I can clean up this code?

A big bowl of spaghetti code.

How can you become a Ruby on Rails expert if the code you read every day stinks? 🦡

Can you learn and apply
best practices
when you’re trapped in a
big ball of stale legacy code?

Can you learn and apply
best practices
when you’re trapped in a
big ball of stale legacy code?

Yes, you can!

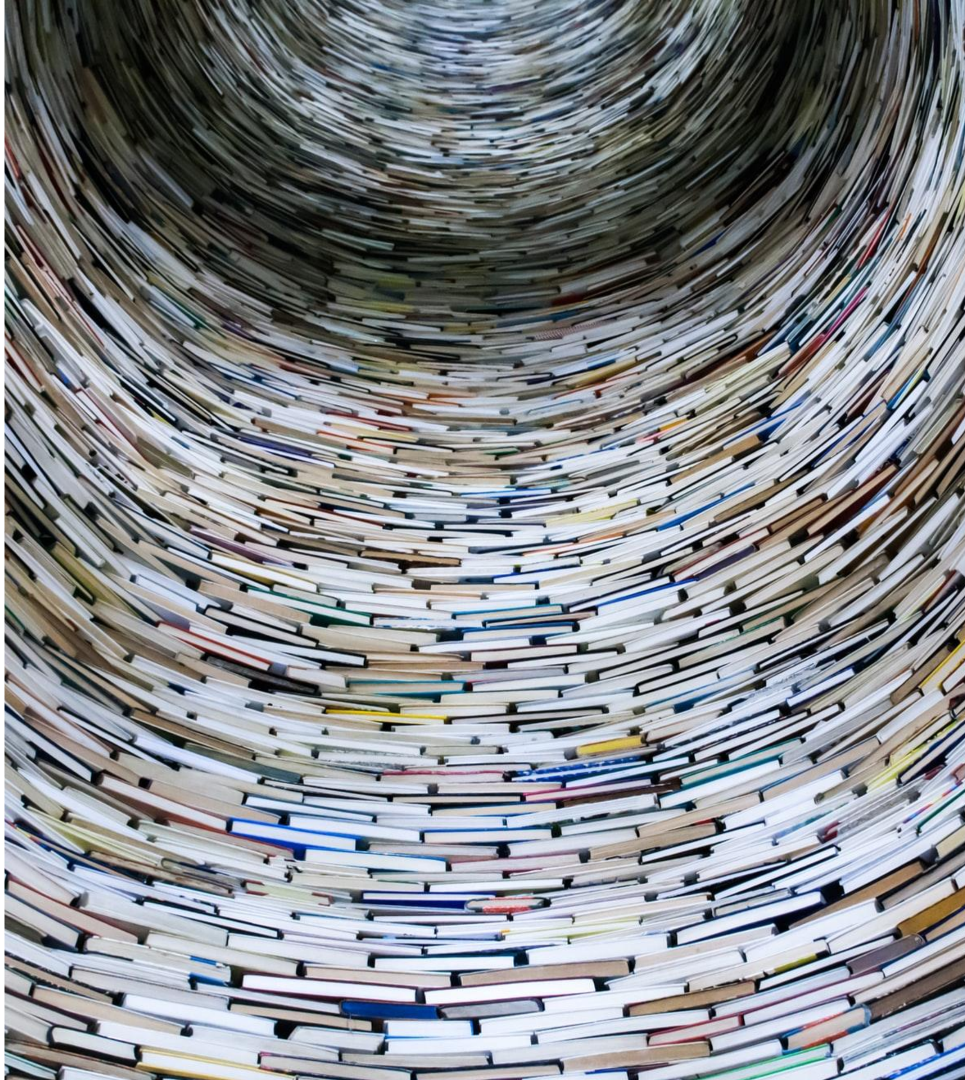
A daily bowl of spaghetti code 🍝

won’t teach you how to write clean code.

If you ask people:

“How do I learn the best practices?”

they will give you 20 different books to read.



They will start throwing these terms at you:

- Skinny controllers, fat models!
- Service Objects!
- SOLID!
- DRY!

What are you supposed to do with all of that?

You want to learn this stuff today to help you fix the mess as soon as possible.

Not in 6 months...

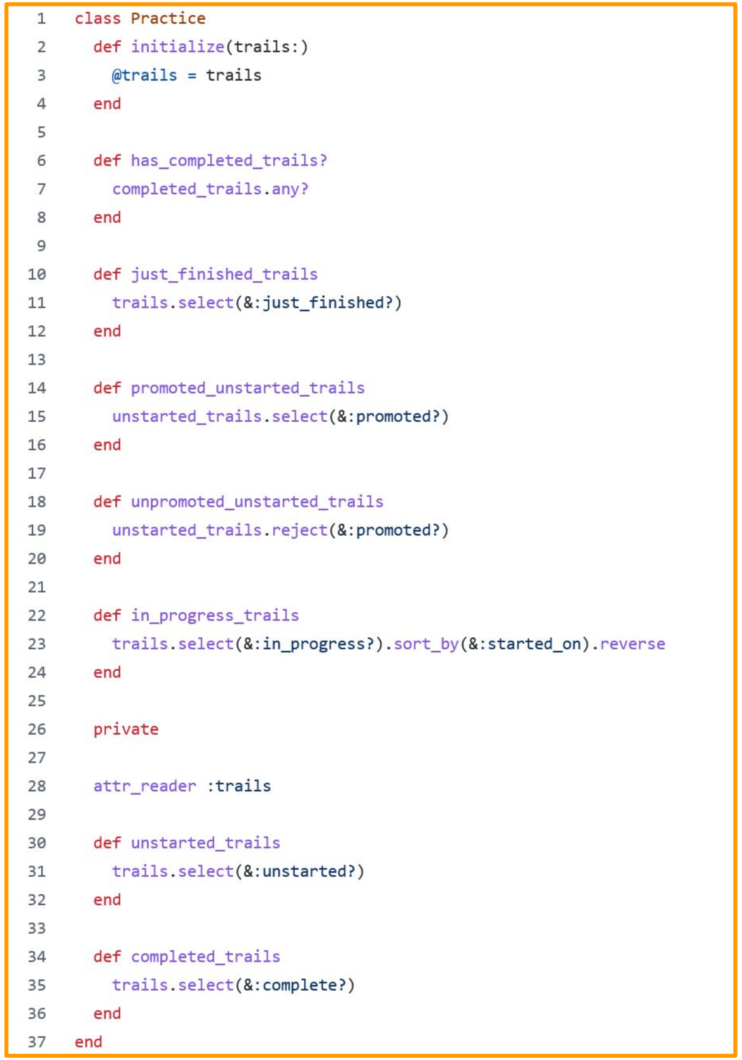
What if you could learn and practice daily with **ONE 10-MINUTES EXERCISE**?

Exercise

1. Set a timer for 10 minutes.
2. Open this link in another tab: practice.rb from Upcase by Thoughtbot
3. Read the app/services/practice.rb class.
4. Ask yourself: What’s going on here?
5. Answer some questions when the time is up! ⌛

Exercise

1. Set a timer for 10 minutes.
2. Open this link in another tab: practice.rb from Upcase by Thoughtbot
3. Read the app/services/practice.rb class.
4. Ask yourself: What's going on here?
5. Answer some questions when the time is up! ⌛



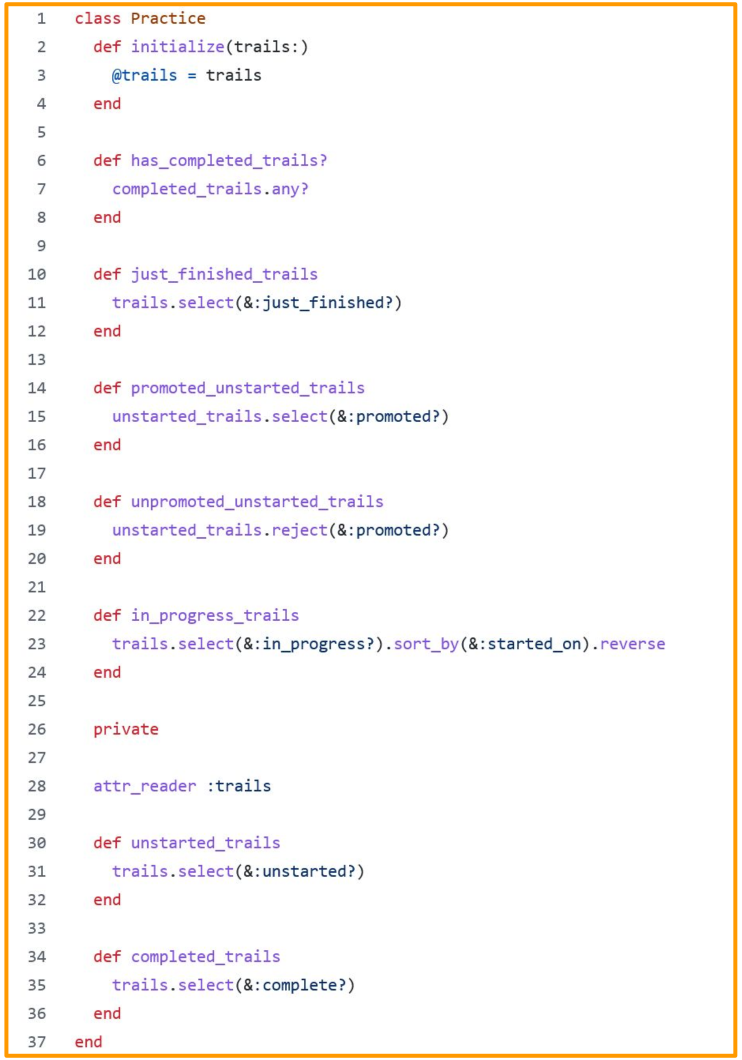
```ruby
class Practice
  def initialize(trails:)
    @trails = trails
  end

  def has_completed_trails?
    completed_trails.any?
  end

  def just_finished_trails
    trails.select(&:just_finished?)
  end

  def promoted_unstarted_trails
    unstarted_trails.select(&:promoted?)
  end

  def unpromoted_unstarted_trails
    unstarted_trails.reject(&:promoted?)
  end

  def in_progress_trails
    trails.select(&:in_progress?).sort_by(&:started_on).reverse
  end

  private

  attr_reader :trails

  def unstarted_trails
    trails.select(&:unstarted?)
  end

  def completed_trails
    trails.select(&:complete?)
  end
end
```

Questions

Times up ⌛! Answer at least one of these questions:

- Why is this class so short? It's less than a hundred lines!
- Why is the variable trails being passed down to the initialize constructor?
- What is this class responsible for? 🤔
- What else picked your interest?

Write down the questions and your answers to make them stick.



```ruby
class Practice
  def initialize(trails:)
    @trails = trails
  end

  def has_completed_trails?
    completed_trails.any?
  end

  def just_finished_trails
    trails.select(&:just_finished?)
  end

  def promoted_unstarted_trails
    unstarted_trails.select(&:promoted?)
  end

  def unpromoted_unstarted_trails
    unstarted_trails.reject(&:promoted?)
  end

  def in_progress_trails
    trails.select(&:in_progress?).sort_by(&:started_on).reverse
  end

  private

  attr_reader :trails

  def unstarted_trails
    trails.select(&:unstarted?)
  end

  def completed_trails
    trails.select(&:complete?)
  end
end
```

Practice every day 🧠

- Repeat this exercise tomorrow, but pick a different class from the same repository.
- Add a daily reminder to your calendar so you don’t forget about it. 📅
- Or do it just after lunch or coffee. Build a new habit!

Why does this work? 🧠

It’s okay if you don’t understand something. Answer *“I don’t know”* and take notes. Use them as a guide on what to focus on next.

The point of this exercise is to help you start seeing some patterns.

It will help you absorb some good practices.

You will see how other experts structure a project to make the code look neat and clean.

Why pick a well-written codebase?

You have to be exposed to good code if you want to advance your skills.

Especially if you’re working on a codebase with bad examples and full of code smells.

Why pick a well-written codebase?

If you get stuck, ask for help.

If you get tired of one codebase, pick another one.

Try to apply these new ideas and patterns in your own work.

Keep practicing and you will learn something useful every day.

Questions?

Questions and feedback: @thdaraujo

More exercises and source material: thd.codes

Join our events for developers: hexdevs.com