Exercises, Tips and Solutions

Chapter 1

Follow the instructions from the Rails on Rack section and replace Rack.application.call with PostsController.action(:index).call.

Chapter 2

To show top-10 Ruby files according to the Flog scoring, you can use the following one-liner:

find app -name "\*.rb" | while read file; do echo $file `flog -s $file | grep ": flog total" | cut -d: -f1`; done | sort -k 2 -nr | head

And now we can combine it with the churn calculation as follows:

find app -name "\*.rb" | while read file; do churn=$(git log --format=oneline -- $file | wc -l); complexity=$(flog -s $file | grep ": flog total" | cut -d: -f1); echo $file `echo "$churn \* $complexity" | bc -l`; done | sort -k 2 -nr | head

Chapter 3

Here are some recommendations on how to re-implement the inline adapter for Active Job:

* Follow Rails naming conventions, so it would be possible to activate the adapter by specifying config.active\_job.queue\_adapter = :custom\_inline.
* Don’t forget to perform arguments serialization. Even though we execute jobs inline, we must ensure fresh objects are passed to #perform. Look at the Job#serialize and ActiveJob::Base.execute methods.
* For delayed execution, you may consider using Ruby Threads.

Chapter 9

Here are some thoughts on how we can integrate the #field\_allowed? method into form objects. Below, we assume that Action Policy is being used.

We can map field\_allowed?(:x) to the corresponding Action Policy invocation:

def field\_allowed?(field\_name)

view\_policy.allowed\_to?(:"#{field\_name}\_field?")

end

We can use the form class name to infer the policy class, i.e., from SearchForm to SearchViewPolicy:

def view\_policy

@view\_policy ||= ActionPolicy.lookup(:"#{model\_name.param\_key}\_view")

end

In the preceding snippet, we use the ActionPolicy.lookup method to instantiate a policy object by its name. We use the \_view suffix to distinguish regular policies from view policies.

All that’s left is to create a SearchViewPolicy class.

Chapter 10

1. You can find the complete example on GitHub: <https://github.com/PacktPublishing/Layered-Design-for-Ruby-on-Rails-Applications/blob/main/Chapter10/03-pluggable-delivery-sketch.rb>.
2. An example implementation can be found in the Active Delivery documentation: https://github.com/palkan/active\_delivery.

Chapter 12

A node pattern for the Lint/RailsCredentials cop may look as follows:

def\_node\_matcher :rails\_credentianls?, <<~PATTERN

(send

(send {(const nil? :Rails) (const (cbase) :Rails)} :application)

:credentials

)

PATTERN

The rest of the cop class will be almost the same as for the Lint/RailsEnv rule.