LAVANDA

Hanami

A(nother) Rack-based
Opinionated Framework

Panos Matsinopoulos
Senior Software Engineer @ Lavanda

LRUG June 2020

LAVANDA

Source: https://en.wikipedia.org/wiki/Hanami

Quick introduction to Lavanda

LAVANDA

Lavanda is a technology platform powering the convergence of residential real estate, hospitality and travel.

Our SaaS toolkit aims to shape the future of:

- multifamily / build-to-rent (BTR)
- serviced apartments
- student housing
- vacation rentals

LAVANDA

Quick introduction to my team

We are 11 s/w engineers in the Lavanda Product team, including:

- 8 Rubyists (some more full-stack than others :))
- 3 dedicated frontend experts, using Angular/React/Typescript

And…

…unfortunately, we're not hiring at the moment! :-)

That's not what brought us here today!

LAVANDA

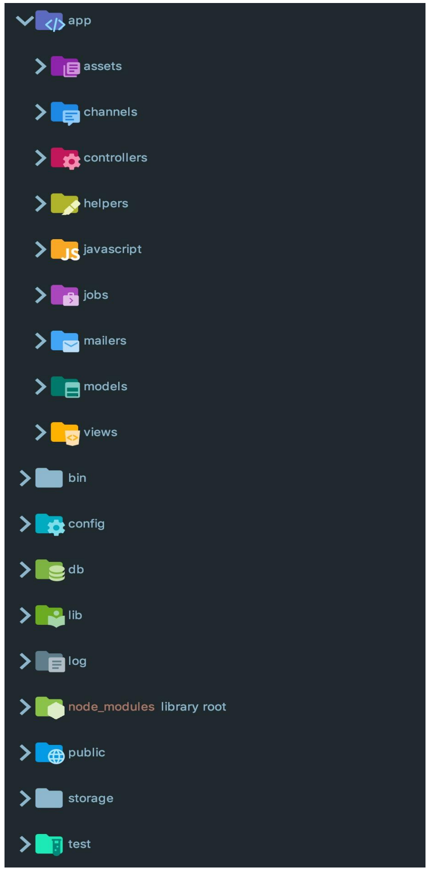
VS

RAILS

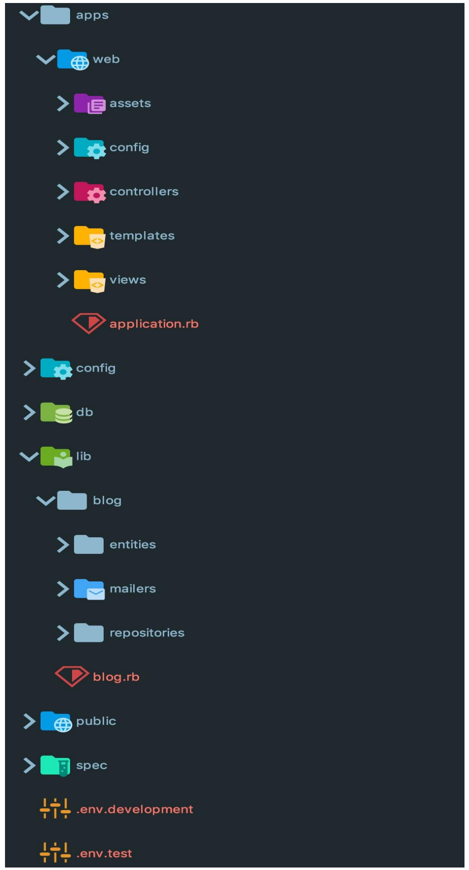
Default Project Folder Structure

RAILS

LAVANDA



```
app
  assets
  channels
  controllers
  helpers
  javascript
  jobs
  mailers
  models
  views
bin
config
db
lib
log
node_modules  library root
public
storage
test
```



```
apps
  web
    assets
    config
    controllers
    templates
    views
    application.rb
config
db
lib
  blog
    entities
    mailers
    repositories
    blog.rb
public
spec
.env.development
.env.test
```

Controllers

- In Rails we have one controller class implementing multiple actions

- Classes that expose a single public method named `call`.
- Each class implements an action.
- Otherwise, we have a lot of similarities. Like for example the `before` and `after` action hooks.

C

LAVANDA

Views vs Templates

- Views: Rails has `*.erb` files that are the HTML content to be returned.
- One per controller action.
- Combined with layout `*.erb` files

- Templates. Again `*.erb` files.
- One per controller action.
- And you can combine with layout templates.
- Views, which are classes preparing data for the templates. One per action.

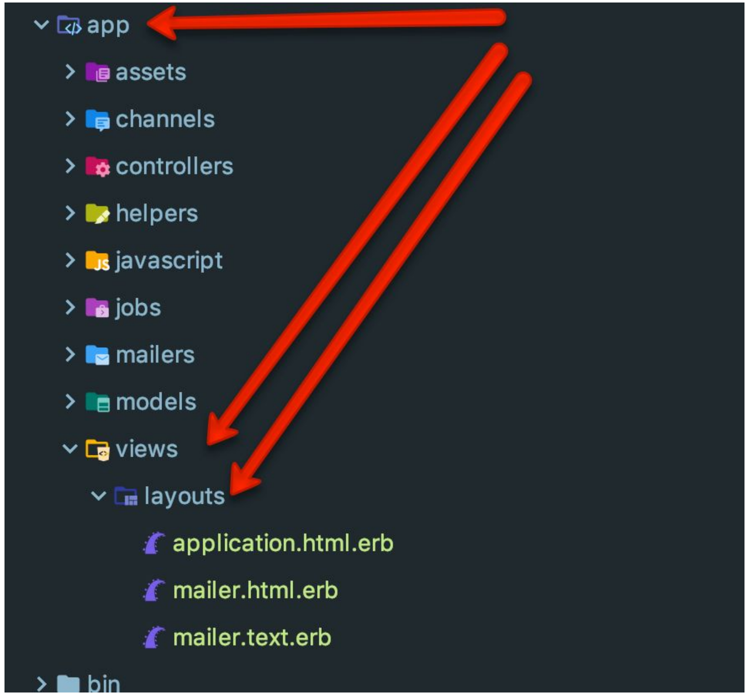
VC

RAILS

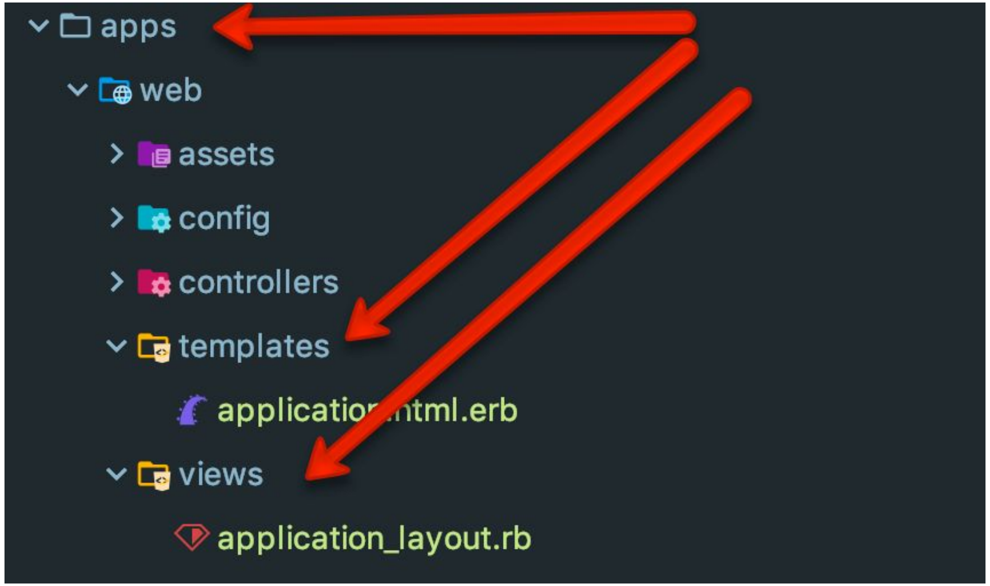
VTC

LAVANDA

Views vs Templates Folders



```
app
  assets
  channels
  controllers
  helpers
  javascript
  jobs
  mailers
  models
  views
    layouts
      application.html.erb
      mailer.html.erb
      mailer.text.erb
bin
```



```
apps
  web
    assets
    config
    controllers
    templates
      application.html.erb
    views
      application_layout.rb
```

Models vs Entities

LAVANDA

- Models deriving from `ActiveRecord::Base`
- Feature/Functional heavy classes

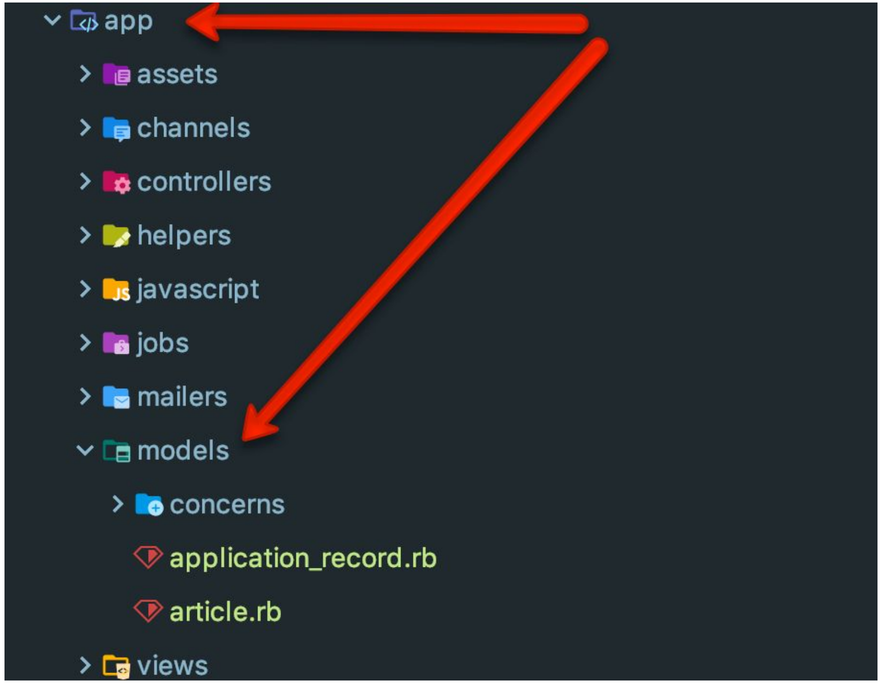
- Entities deriving from `Hanami::Entity`
- Light weight classes
- By default they have an id and you can read the value of its properties. But you can't change their values.

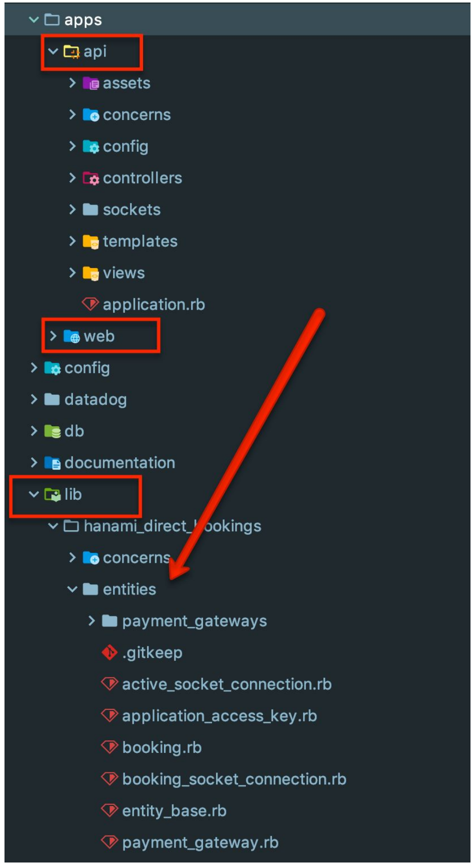
MVC

RAILS

EVTC

Models vs Entities Folders


```
app
  assets
  channels
  controllers
  helpers
  javascript
  jobs
  mailers
  models
    concerns
    application_record.rb
    article.rb
  views
```


```
apps
  api
    assets
    concerns
    config
    controllers
    sockets
    templates
    views
    application.rb
  web
config
datadog
db
documentation
lib
  hanami_direct_bookings
    concerns
    entities
      payment_gateways
      .gitkeep
      active_socket_connection.rb
      application_access_key.rb
      booking.rb
      booking_socket_connection.rb
      entity_base.rb
      payment_gateway.rb
```

LAVANDA

RAILS

Entities

- Derive from `Hanami::Entity`
- They have an `id`.
- And they are read-only.
- They are updated via Repositories.

ActiveRecord vs Repository Pattern

LAVANDA

[ActiveRecord] An object that wraps a row in a database table or view, encapsulates the database access, and adds domain logic on that data. (Martin Fowler)

[A Repository] mediates between the domain and data mapping layers using a collection-like interface for accessing domain objects (Martin Fowler)

Update or Delete an Entity

LAVANDA

- `BookingRepository.new.update(id, data)`
- `BookingRepository.new.delete(id)`

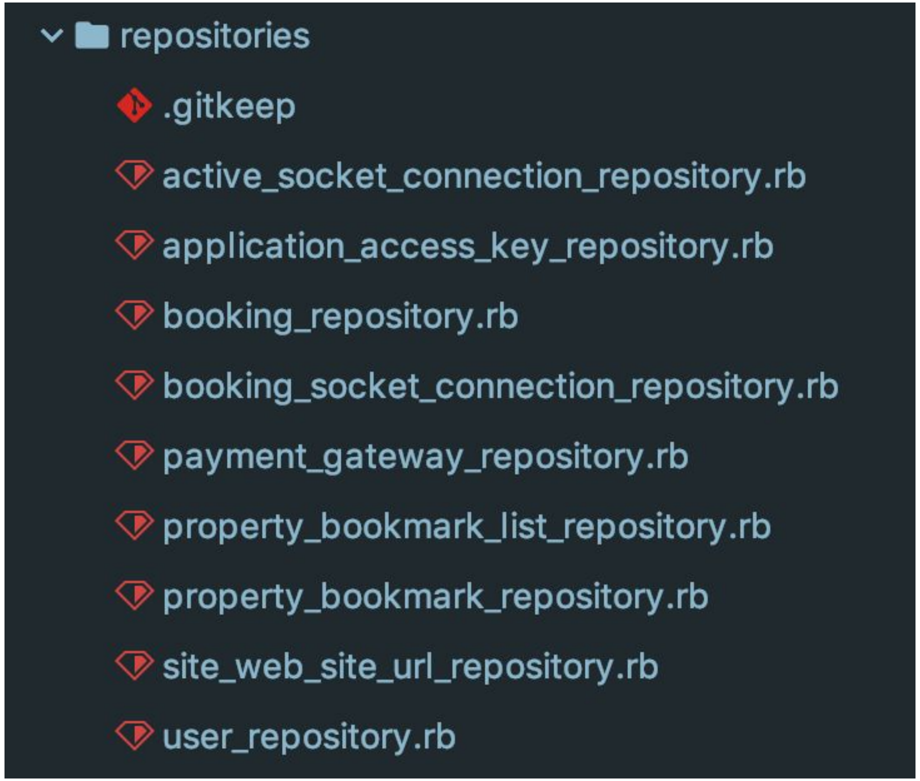
Likes vs Doesn't Like

LAVANDA

- Hanami doesn't like this:
  - `Book.where(author_id: 23).order(:published_at).limit(8)`
- But, it likes this:
  - `BookRepository.new.most_recent_by_author(author, limit: 8)`

Example Repositories folder

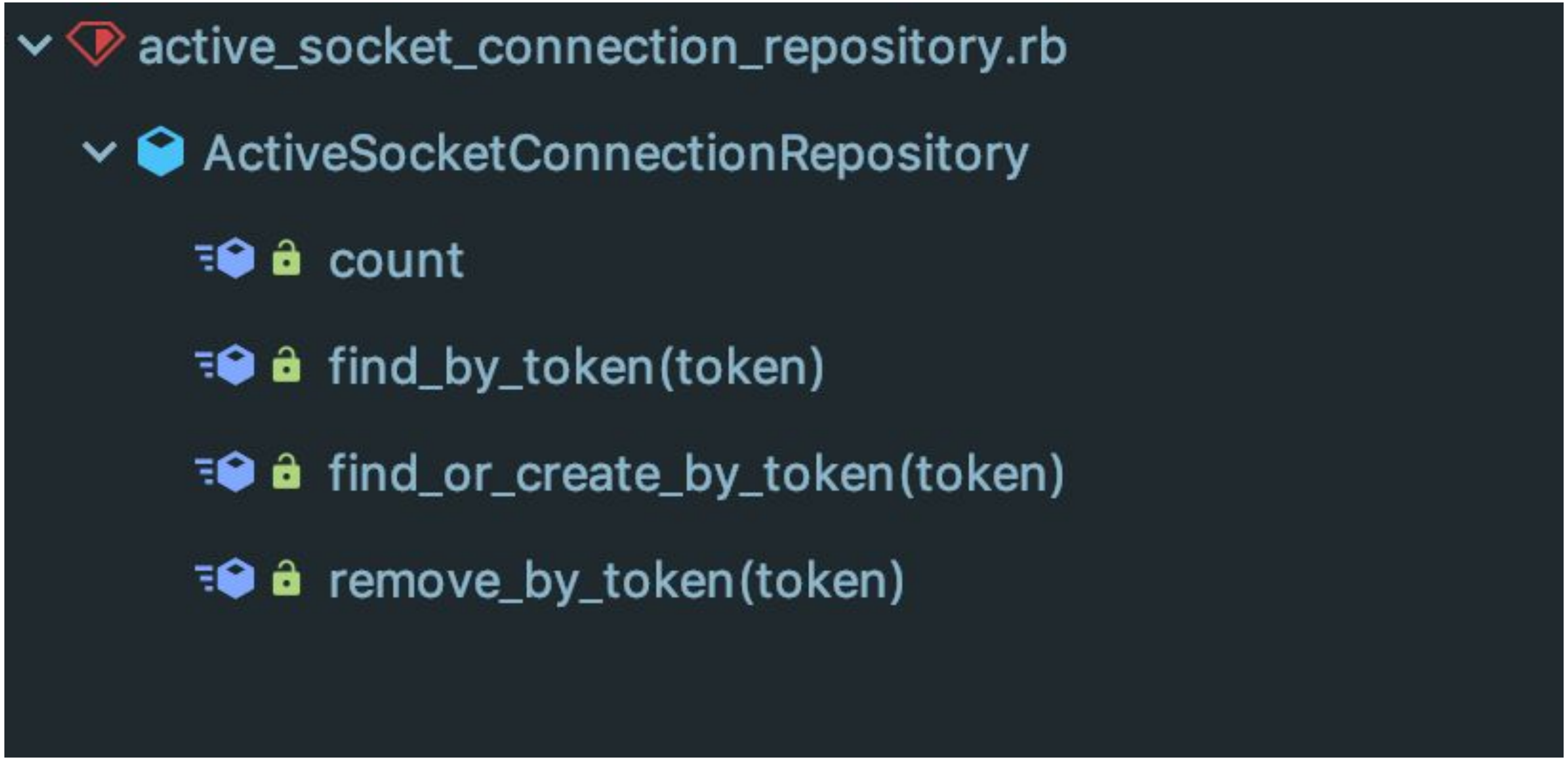
LAVANDA



```
repositories
    .gitkeep
    active_socket_connection_repository.rb
    application_access_key_repository.rb
    booking_repository.rb
    booking_socket_connection_repository.rb
    payment_gateway_repository.rb
    property_bookmark_list_repository.rb
    property_bookmark_repository.rb
    site_web_site_url_repository.rb
    user_repository.rb
```

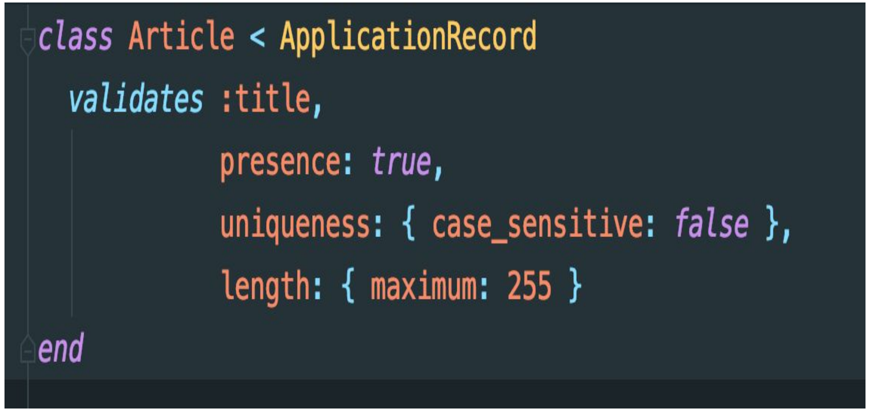
Repository: What’s inside

LAVANDA

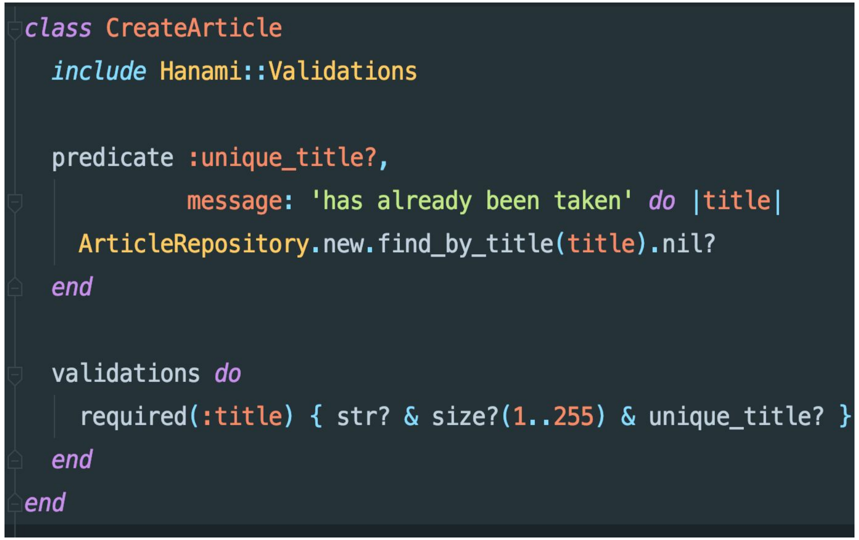


```
active_socket_connection_repository.rb
  ActiveSocketConnectionRepository
    count
    find_by_token(token)
    find_or_create_by_token(token)
    remove_by_token(token)
```

Validations



```ruby
class Article < ApplicationRecord
  validates :title,
            presence: true,
            uniqueness: { case_sensitive: false },
            length: { maximum: 255 }
end
```

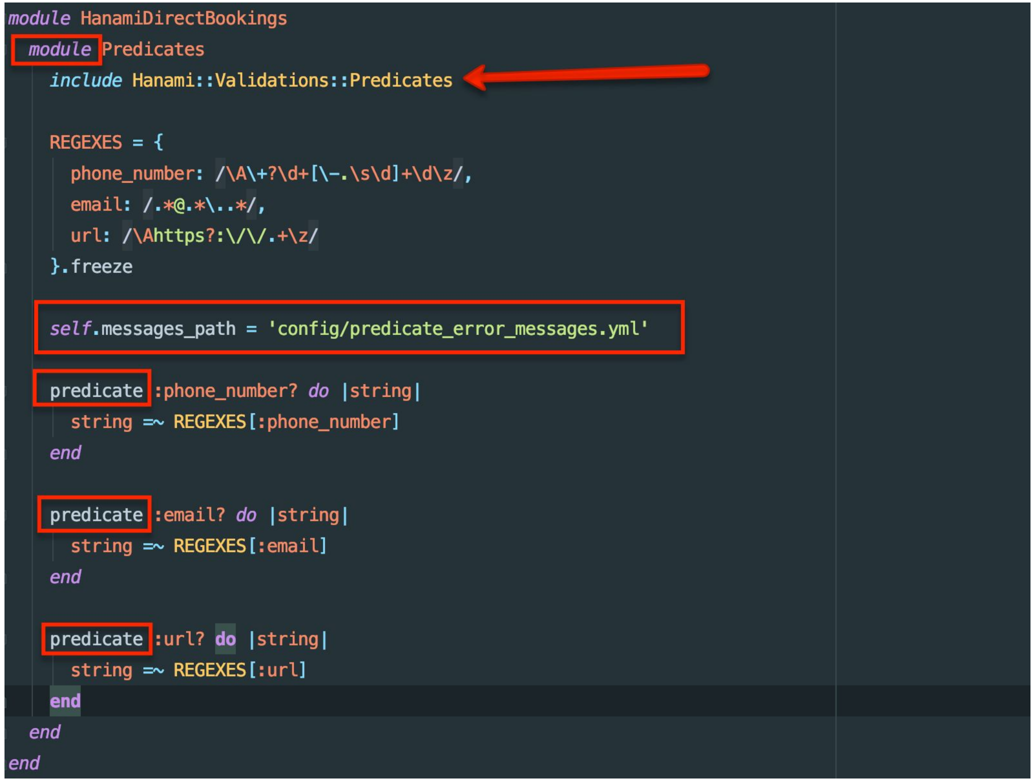


```ruby
class CreateArticle
  include Hanami::Validations

  predicate :unique_title?,
            message: 'has already been taken' do |title|
    ArticleRepository.new.find_by_title(title).nil?
  end

  validations do
    required(:title) { str? & size?(1..255) & unique_title? }
  end
end
```

LRUG June 2020

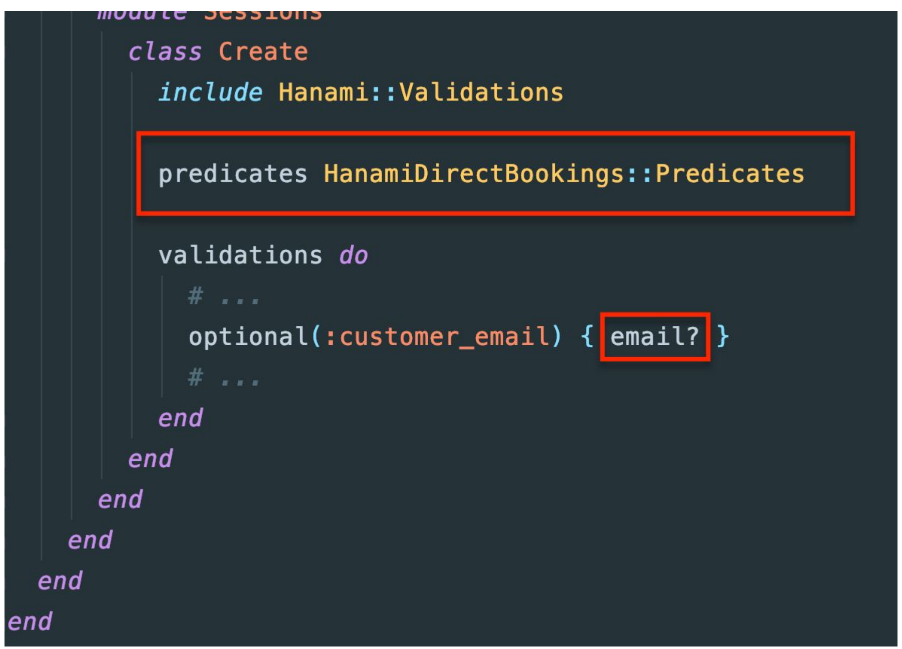
Global Custom Validation Predicates

LAVANDA

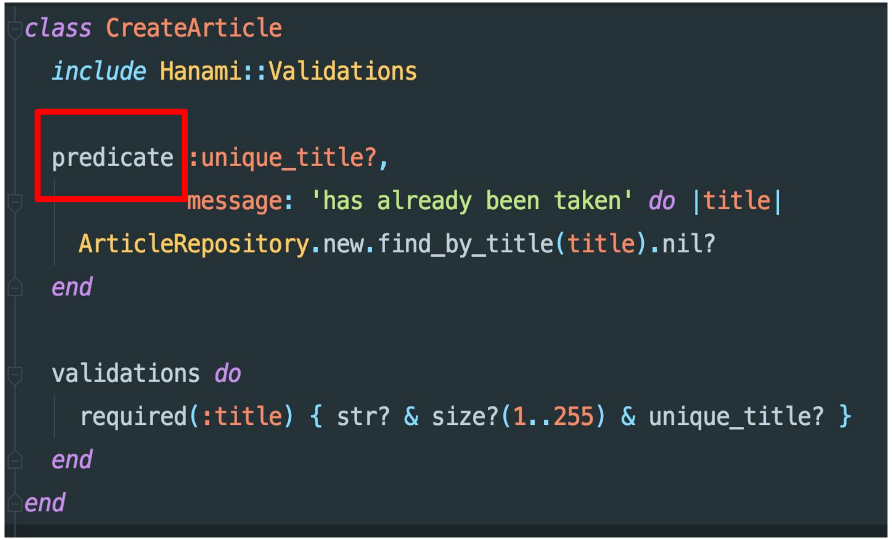


```ruby
module HanamiDirectBookings
  module Predicates
    include Hanami::Validations::Predicates

    REGEXES = {
      phone_number: /\A\+?\d+[\-.\s\d]+\d\z/,
      email: /.*@.*\..*/,
      url: /\Ahttps?:\/\/.+\z/
    }.freeze

    self.messages_path = 'config/predicate_error_messages.yml'

    predicate :phone_number? do |string|
      string =~ REGEXES[:phone_number]
    end

    predicate :email? do |string|
      string =~ REGEXES[:email]
    end

    predicate :url? do |string|
      string =~ REGEXES[:url]
    end
  end
end
```

Using Global Predicates vs Inline Ones



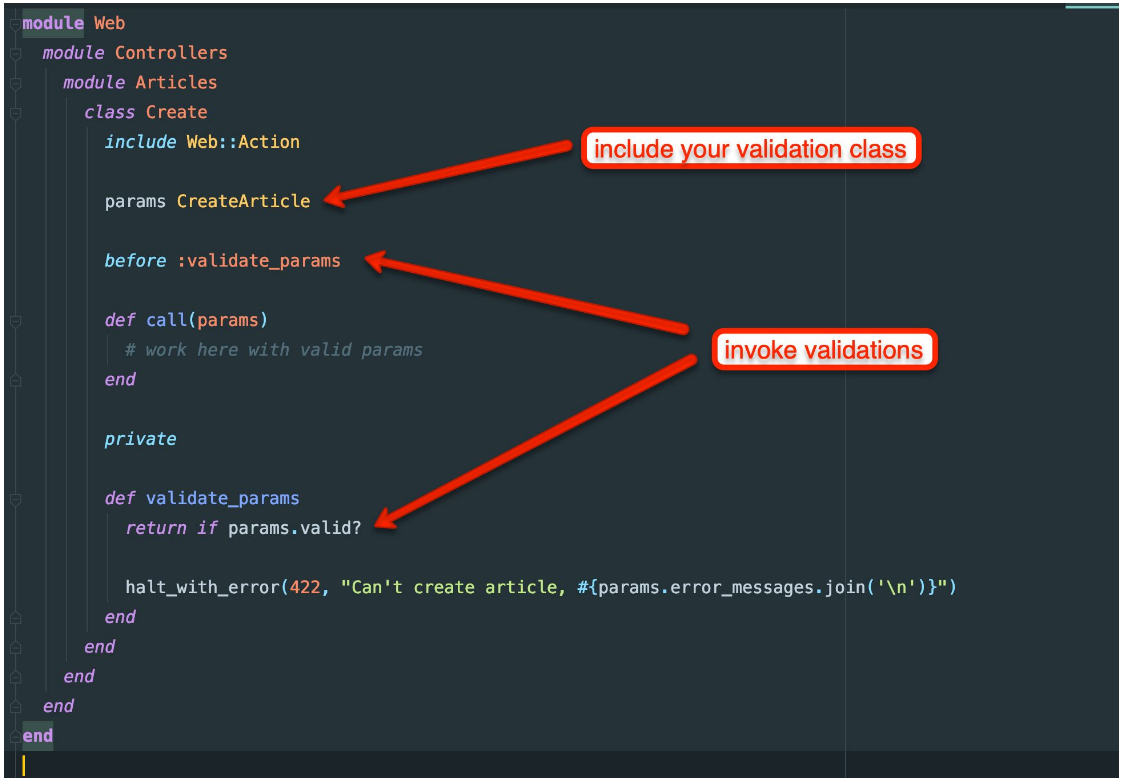
```
      module Sessions
        class Create
          include Hanami::Validations

          predicates HanamiDirectBookings::Predicates

          validations do
            # ...
            optional(:customer_email) { email? }
            # ...
          end
        end
      end
    end
  end
end
```



```
class CreateArticle
  include Hanami::Validations

  predicate :unique_title?,
            message: 'has already been taken' do |title|
    ArticleRepository.new.find_by_title(title).nil?
  end

  validations do
    required(:title) { str? & size?(1..255) & unique_title? }
  end
end
```

Invoking Hanami Validations

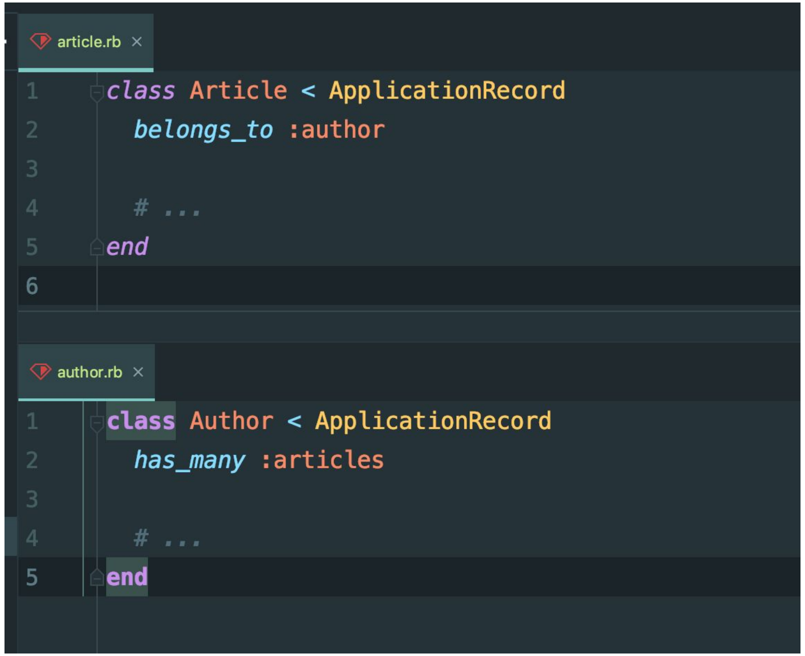
LAVANDA



```ruby
module Web
  module Controllers
    module Articles
      class Create
        include Web::Action

        params CreateArticle

        before :validate_params

        def call(params)
          # work here with valid params
        end

        private

        def validate_params
          return if params.valid?

          halt_with_error(422, "Can't create article, #{params.error_messages.join('\n')}")
        end
      end
    end
  end
end
```

Associations (Rails vs Hanami)

LAVANDA

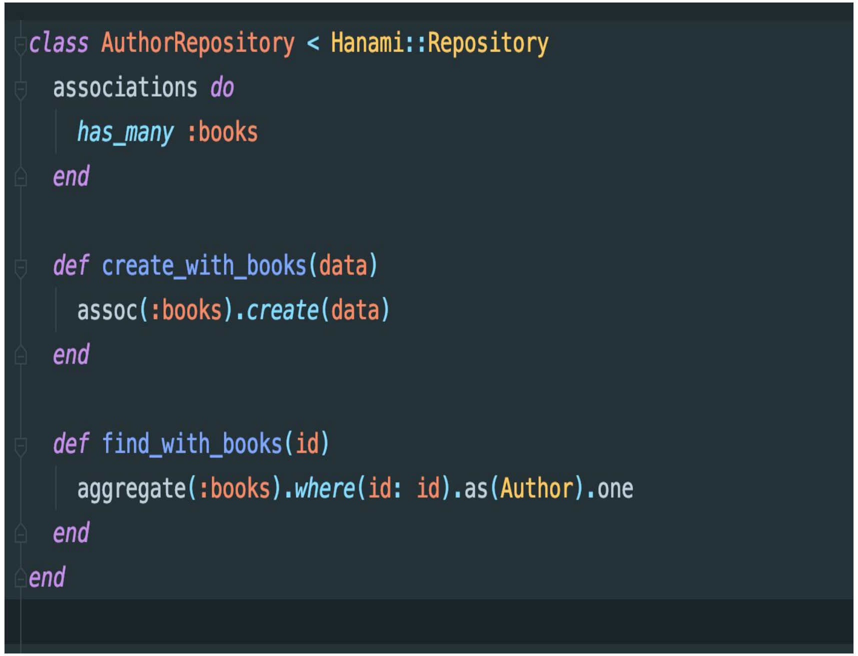


article.rb

```ruby
class Article < ApplicationRecord
  belongs_to :author

  # ...
end

```

author.rb

```ruby
class Author < ApplicationRecord
  has_many :articles

  # ...
end
```



```ruby
class AuthorRepository < Hanami::Repository
  associations do
    has_many :books
  end

  def create_with_books(data)
    assoc(:books).create(data)
  end

  def find_with_books(id)
    aggregate(:books).where(id: id).as(Author).one
  end
end
```

RAILS

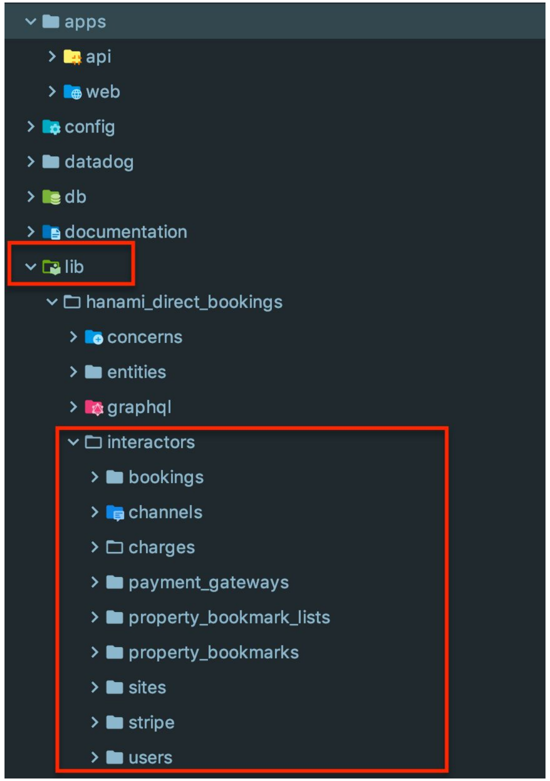
Interactor (optional but useful)

LAVANDA

- More like a Service object
- `Initialize` and `call`
- Can use validations
- They are Hanami-application independent
    - They live in the `lib` folder tree and not in any application-specific one.

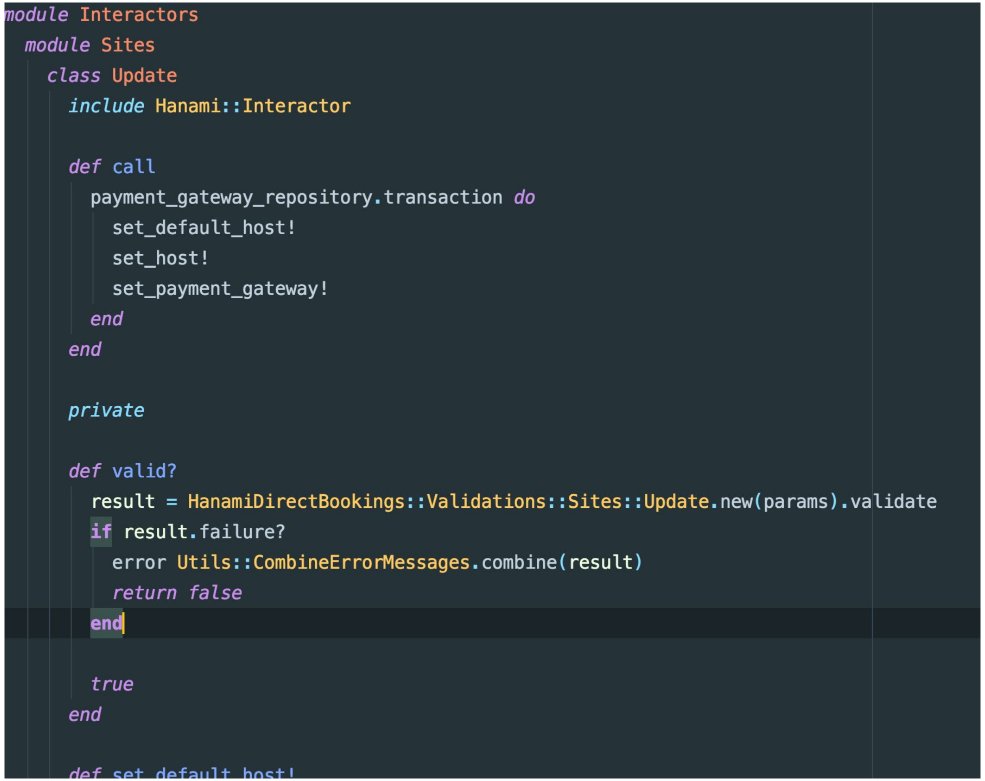
Interactors Live in `lib` Folder

LAVANDA



```
apps
  api
  web
config
datadog
db
documentation
lib
  hanami_direct_bookings
    concerns
    entities
    graphql
    interactors
      bookings
      channels
      charges
      payment_gateways
      property_bookmark_lists
      property_bookmarks
      sites
      stripe
      users
```

Interactor Example

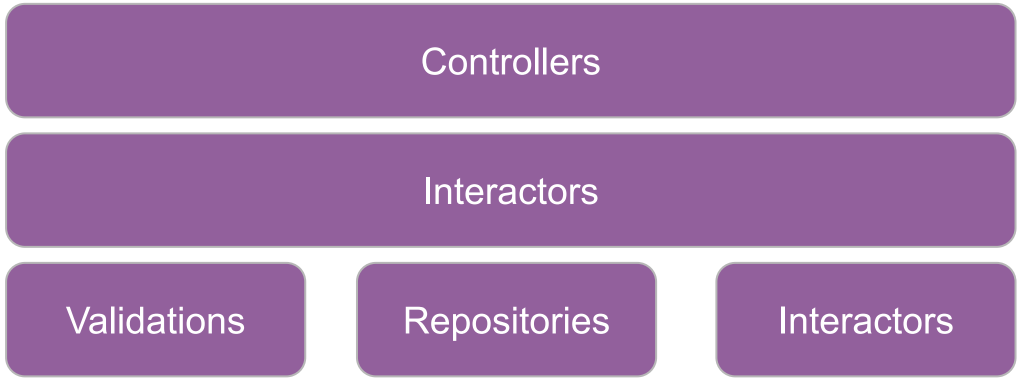


```ruby
module Interactors
  module Sites
    class Update
      include Hanami::Interactor

      def call
        payment_gateway_repository.transaction do
          set_default_host!
          set_host!
          set_payment_gateway!
        end
      end

      private

      def valid?
        result = HanamiDirectBookings::Validations::Sites::Update.new(params).validate
        if result.failure?
          error Utils::CombineErrorMessages.combine(result)
          return false
        end

        true
      end

      def set_default_host!
```

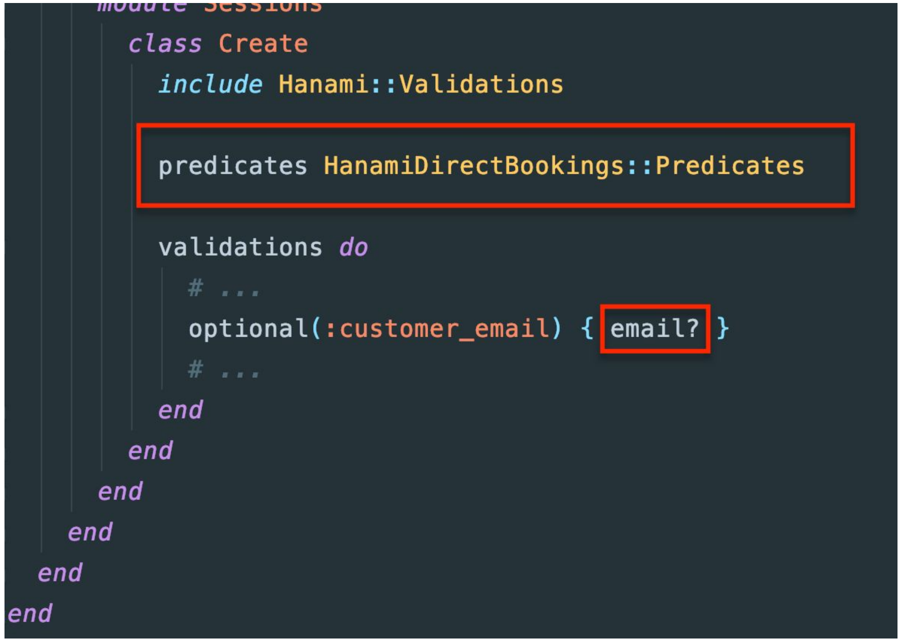
LRUG June 2020

Lavanda Hanami Preferred Architecture

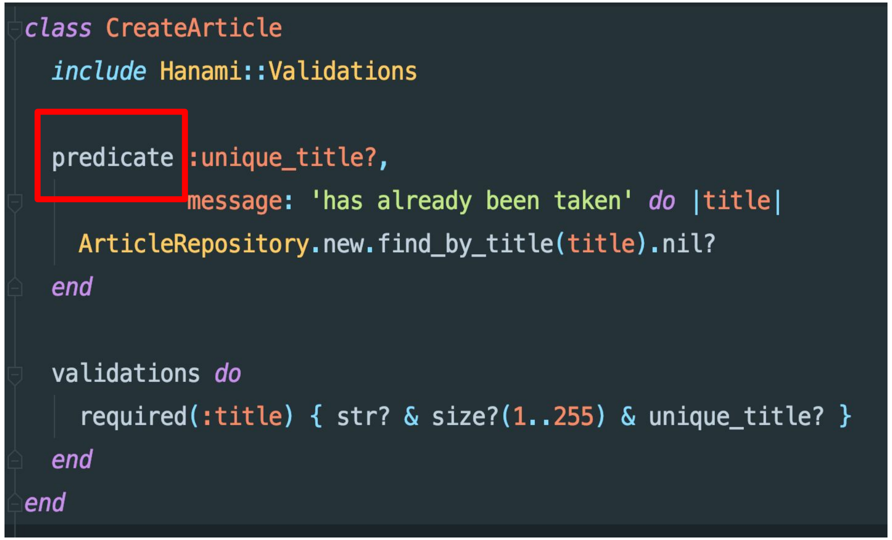
LAVANDA



Lavanda PR to Hanami Validations (1/8)



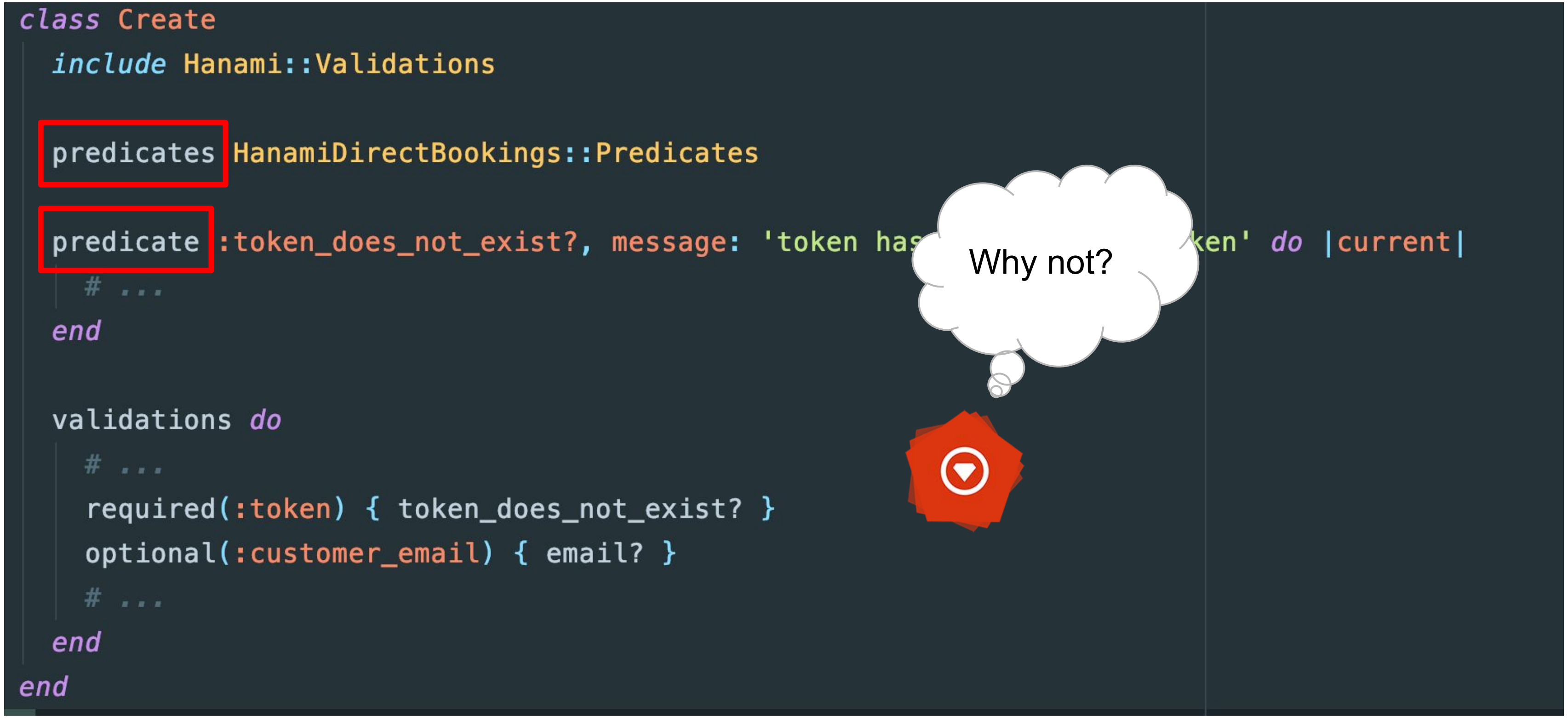
```
      module Sessions
        class Create
          include Hanami::Validations

          predicates HanamiDirectBookings::Predicates

          validations do
            # ...
            optional(:customer_email) { email? }
            # ...
          end
        end
      end
    end
  end
end
```



```
class CreateArticle
  include Hanami::Validations

  predicate :unique_title?,
            message: 'has already been taken' do |title|
    ArticleRepository.new.find_by_title(title).nil?
  end

  validations do
    required(:title) { str? & size?(1..255) & unique_title? }
  end
end
```

Lavanda PR to Hanami Validations (2/8)

LAVANDA



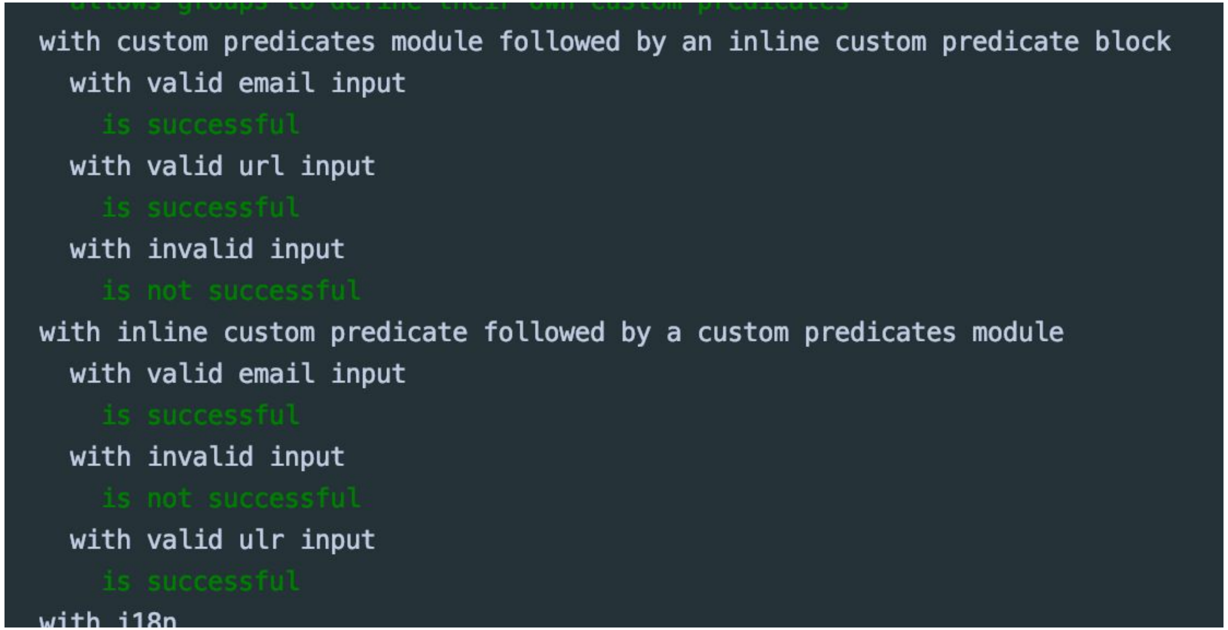
```ruby
class Create
  include Hanami::Validations

  predicates HanamiDirectBookings::Predicates

  predicate :token_does_not_exist?, message: 'token has        ken' do |current|
    # ...
  end

  validations do
    # ...
    required(:token) { token_does_not_exist? }
    optional(:customer_email) { email? }
    # ...
  end
end
```

Lavanda PR to Hanami Validations (3/8)

LAVANDA

- It seems that you can’t use both.
  - Module definition eliminates inline predicate definition
- Let’s create a PR
  - Find the repo: https://github.com/hanami/validations
  - Find the specs: `spec/unit/hanami/validations/predicates/schema/custom_spec.rb`
  - Amend the existing specs.
  - Change the library to comply with new specs.
  - Issue the PR.

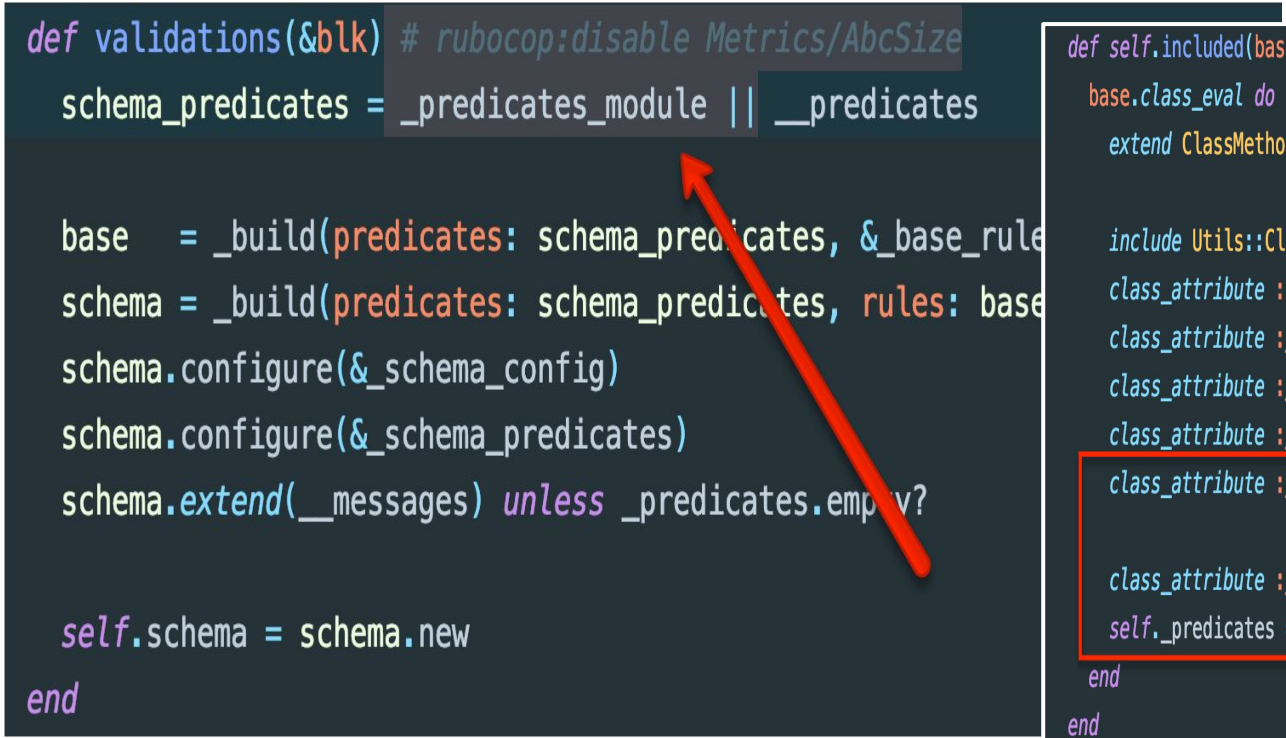
Lavanda PR to Hanami Validations (4/8)



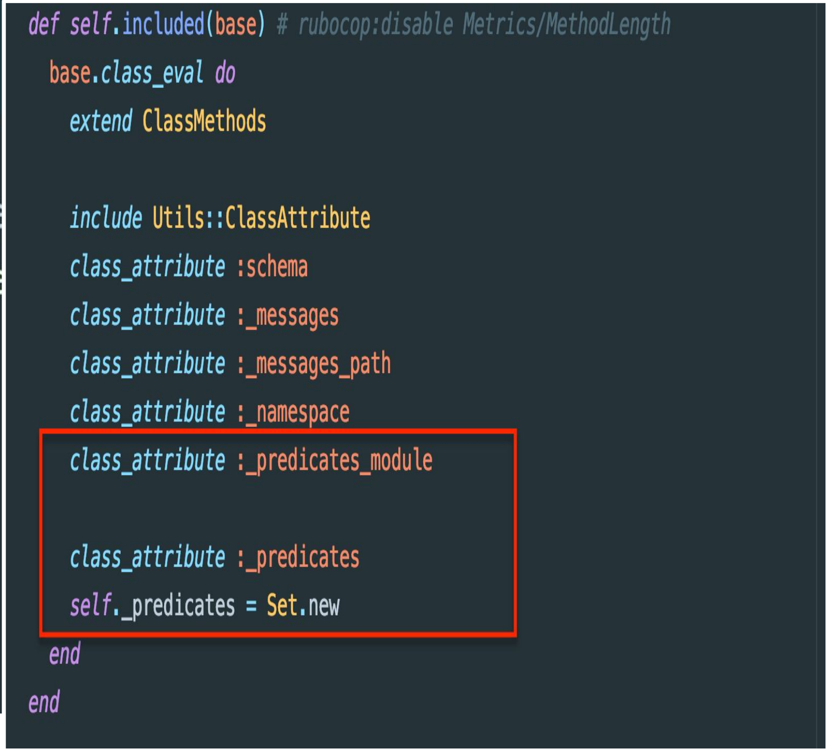
```
  allows groups to define their own custom predicates
with custom predicates module followed by an inline custom predicate block
  with valid email input
    is successful
  with valid url input
    is successful
  with invalid input
    is not successful
with inline custom predicate followed by a custom predicates module
  with valid email input
    is successful
  with invalid input
    is not successful
  with valid ulr input
    is successful
with i18n
```

Lavanda PR to Hanami Validations (5/8)

LAVANDA



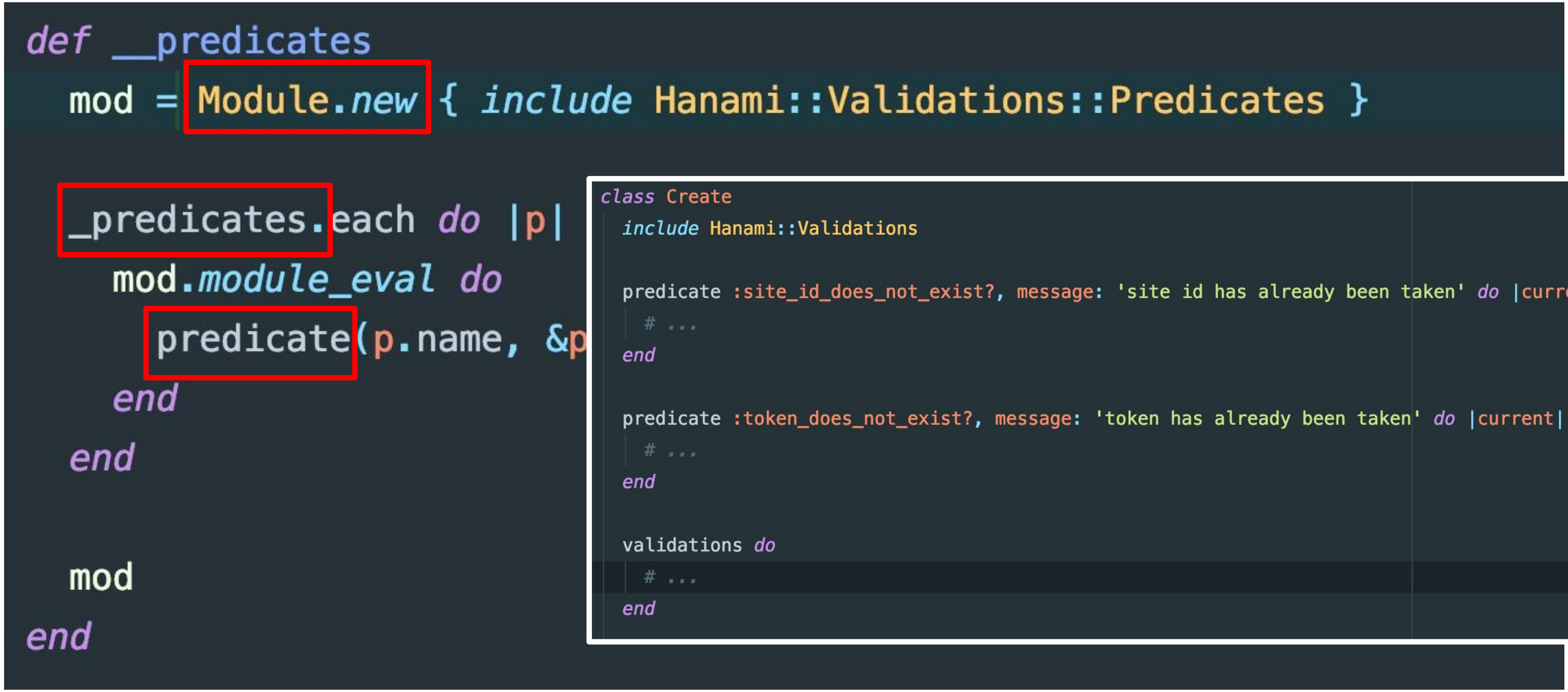
```ruby
def validations(&blk) # rubocop:disable Metrics/AbcSize
  schema_predicates = _predicates_module || __predicates

  base   = _build(predicates: schema_predicates, &_base_rule
  schema = _build(predicates: schema_predicates, rules: base
  schema.configure(&_schema_config)
  schema.configure(&_schema_predicates)
  schema.extend(__messages) unless _predicates.empty?

  self.schema = schema.new
end
```



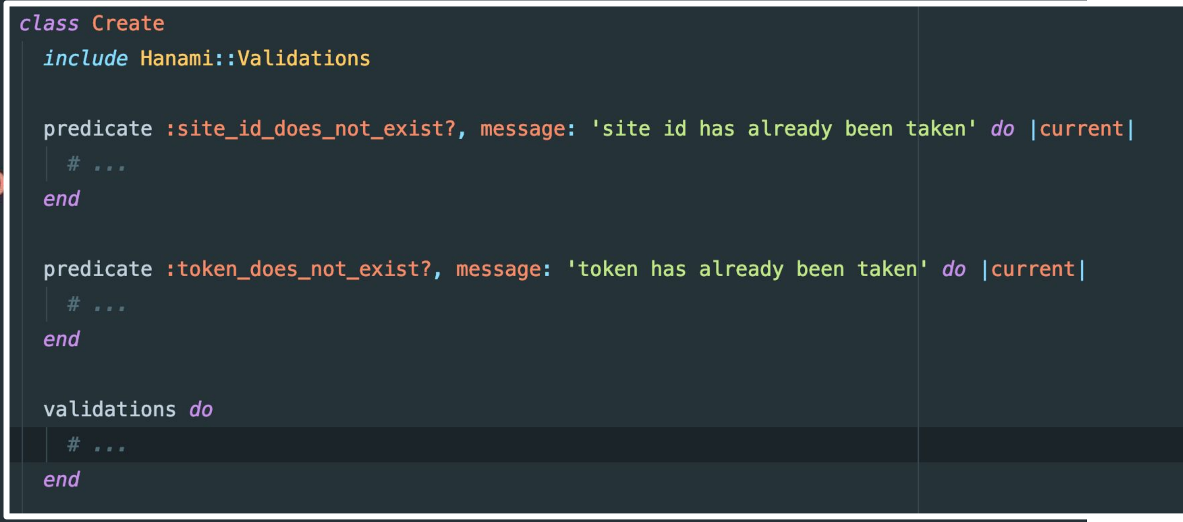
```ruby
def self.included(base) # rubocop:disable Metrics/MethodLength
  base.class_eval do
    extend ClassMethods

    include Utils::ClassAttribute
    class_attribute :schema
    class_attribute :_messages
    class_attribute :_messages_path
    class_attribute :_namespace
    class_attribute :_predicates_module


    class_attribute :_predicates
    self._predicates = Set.new
  end
end
```

Lavanda PR to Hanami Validations (6/8)

LAVANDA



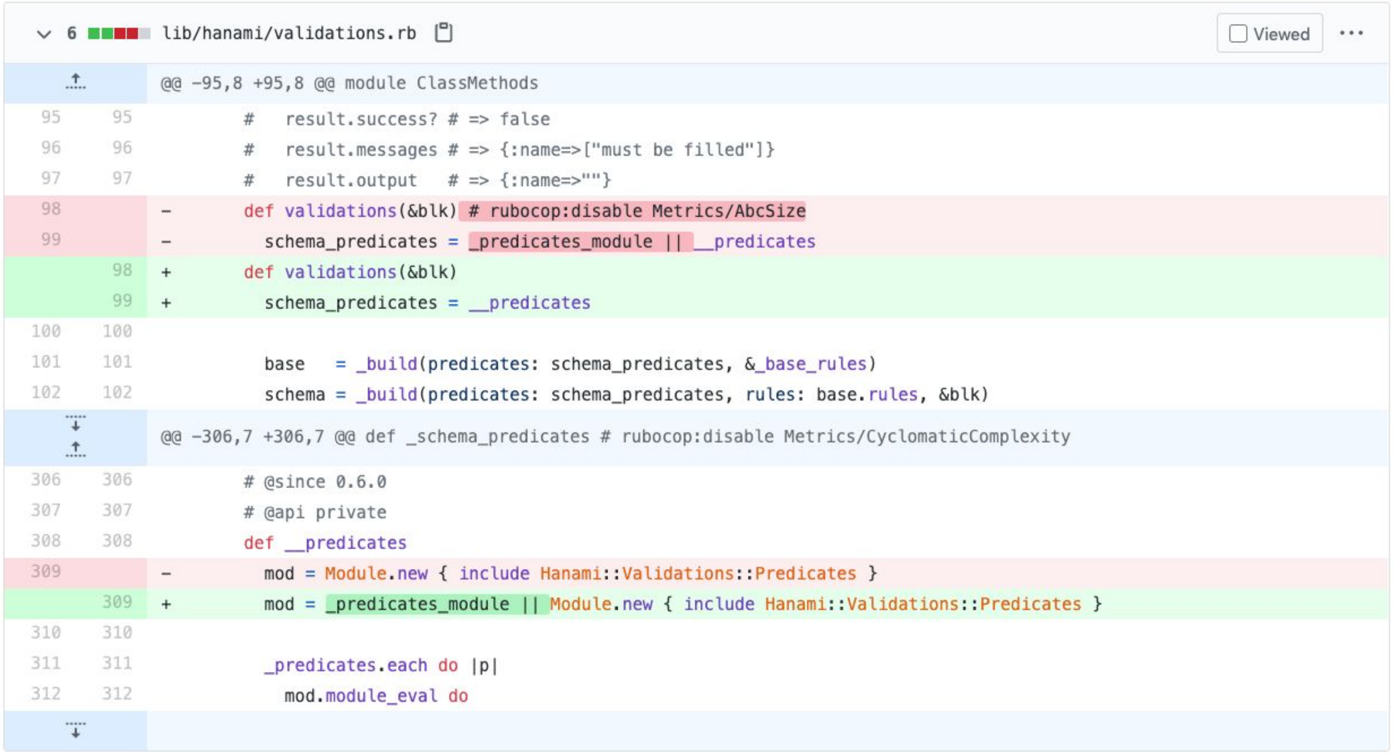
```ruby
def __predicates
  mod = Module.new { include Hanami::Validations::Predicates }

  _predicates.each do |p|
    mod.module_eval do
      predicate(p.name, &p
    end
  end

  mod
end
```



```ruby
class Create
  include Hanami::Validations

  predicate :site_id_does_not_exist?, message: 'site id has already been taken' do |current|
    # ...
  end

  predicate :token_does_not_exist?, message: 'token has already been taken' do |current|
    # ...
  end

  validations do
    # ...
  end
```

Lavanda PR to Hanami Validations (7/8)

LAVANDA



6 lib/hanami/validations.rb

Viewed

```diff
@@ -95,8 +95,8 @@ module ClassMethods
         #   result.success? # => false
         #   result.messages # => {:name=>["must be filled"]}
         #   result.output   # => {:name=>""}
-        def validations(&blk) # rubocop:disable Metrics/AbcSize
-          schema_predicates = _predicates_module || __predicates
+        def validations(&blk)
+          schema_predicates = __predicates

          base   = _build(predicates: schema_predicates, &_base_rules)
          schema = _build(predicates: schema_predicates, rules: base.rules, &blk)
@@ -306,7 +306,7 @@ def _schema_predicates # rubocop:disable Metrics/CyclomaticComplexity
        # @since 0.6.0
        # @api private
        def __predicates
-         mod = Module.new { include Hanami::Validations::Predicates }
+         mod = _predicates_module || Module.new { include Hanami::Validations::Predicates }

          _predicates.each do |p|
            mod.module_eval do
```

Lavanda PR to Hanami Validations (8/8)

LAVANDA

https://github.com/hanami/validations/pull/196

(*) Unfortunately, currently failing due to some misconfiguration on their CI in relation to the rubocop requirements

Websockets Support In Hanami

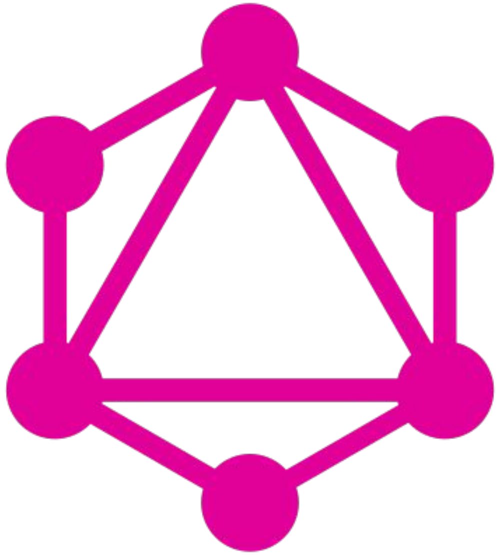
LAVANDA

- Rails has Action Cable.
- Hanami does not offer websockets support out of the box.
- However, we have integrated AnyCable with the help of the gem `litecable`.

Hanami and GraphQL APIs

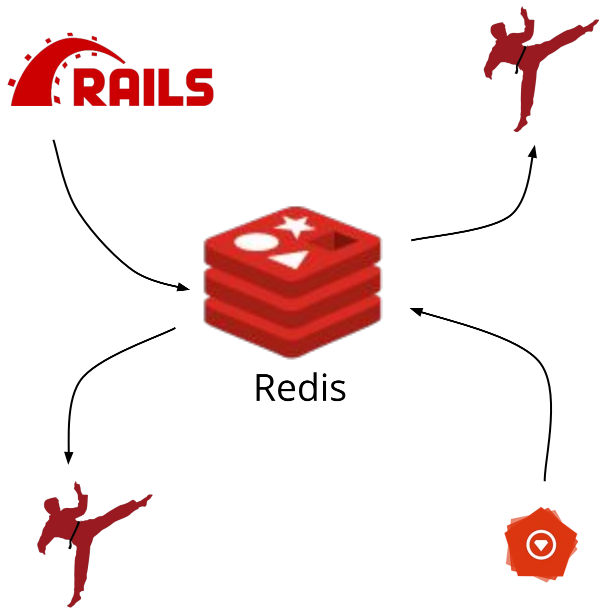
LAVANDA

- Using `graphql` gem



Breaking Monolith

- Server-to-Server using `api-auth`.
- Database-level integration using Background Tasks.
  - Extremely performant
  - Drawback:
    - one server knows the queue name, the class name and `perform` arguments implemented in the other
  - Like an async remote method invocation



```
Sidekiq::Client.push(
 'class' =>   '<class name of remote worker>',
 'args' => [<args of perform>],
 'queue' => '<queue-name>',
 'retry' => false
)
```

Overall Level of Satisfaction

LAVANDA

- We are very satisfied
- Ideal for implementing microservices
- Favors small classes
- It protects you from some common engineering errors
- We would go again with Hanami or
- We would go with Rails, but apply Hanami principles

LAVANDA

Questions and Answers

LAVANDA

Thank you!