ViewComponent

Czy to przyszłość tworzenia widoków w Ruby on Rails?

2N

11.04.2023

O mnie

Mateusz Białowąs

Programista RAILS w 2N

2 lata temu mój szef przyszedł na zajęcia szkieletówki na pb, zeby zachęcić studentów aby pisali projekt w railsach. Pod koniec projektu przyszedłem na staż i już zostałem w tej firmie.

W ostatnim projekcie miałem okazję korzystać z biblioteki view component i dzisiaj chciałbym ją wam pokazać.

Celem prezentacji jest zaprezentowanie biblioteki ViewComponent osobom, które nie miały jeszcze z nią doświadczenia, oraz demonstracja bardziej zaawansowanych zastosowań tej biblioteki dla pozostałych osób.

Agenda

1. Wprowadzenie do ViewComponent
2. Zalety stosowania ViewComponent
3. Generatory i konfiguracja ViewComponent
4. Przekazywanie danych
5. Inne funkcjonalności
6. Dobre praktyki
7. Porównanie wydajności: ViewComponent kontra inne rozwiązania
8. Podsumowanie i sesja pytań i odpowiedzi

Monada to abstrakcyjny pojemnik na wartości, który udostępnia specjalne metody do przetwarzania tej wartości.

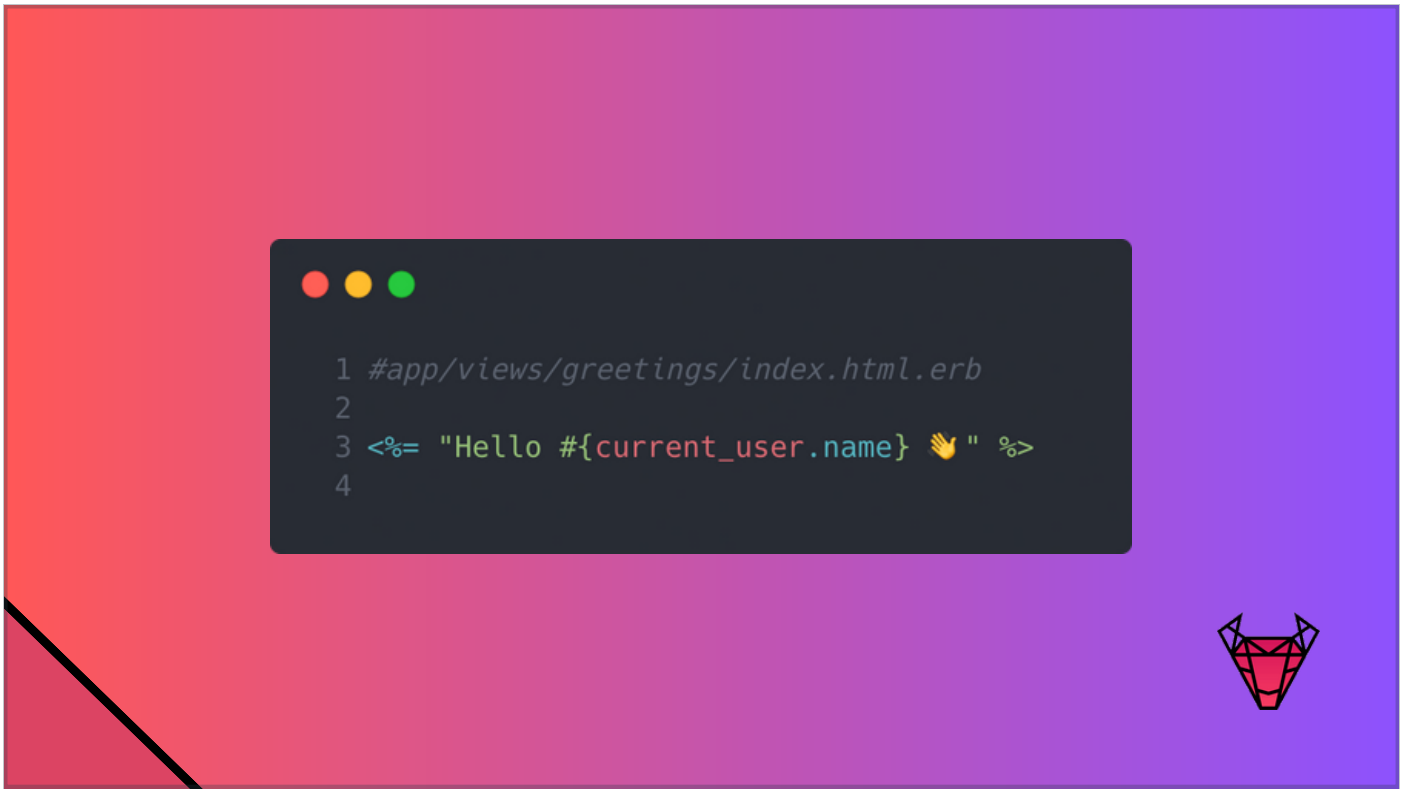
Wprowadzenie

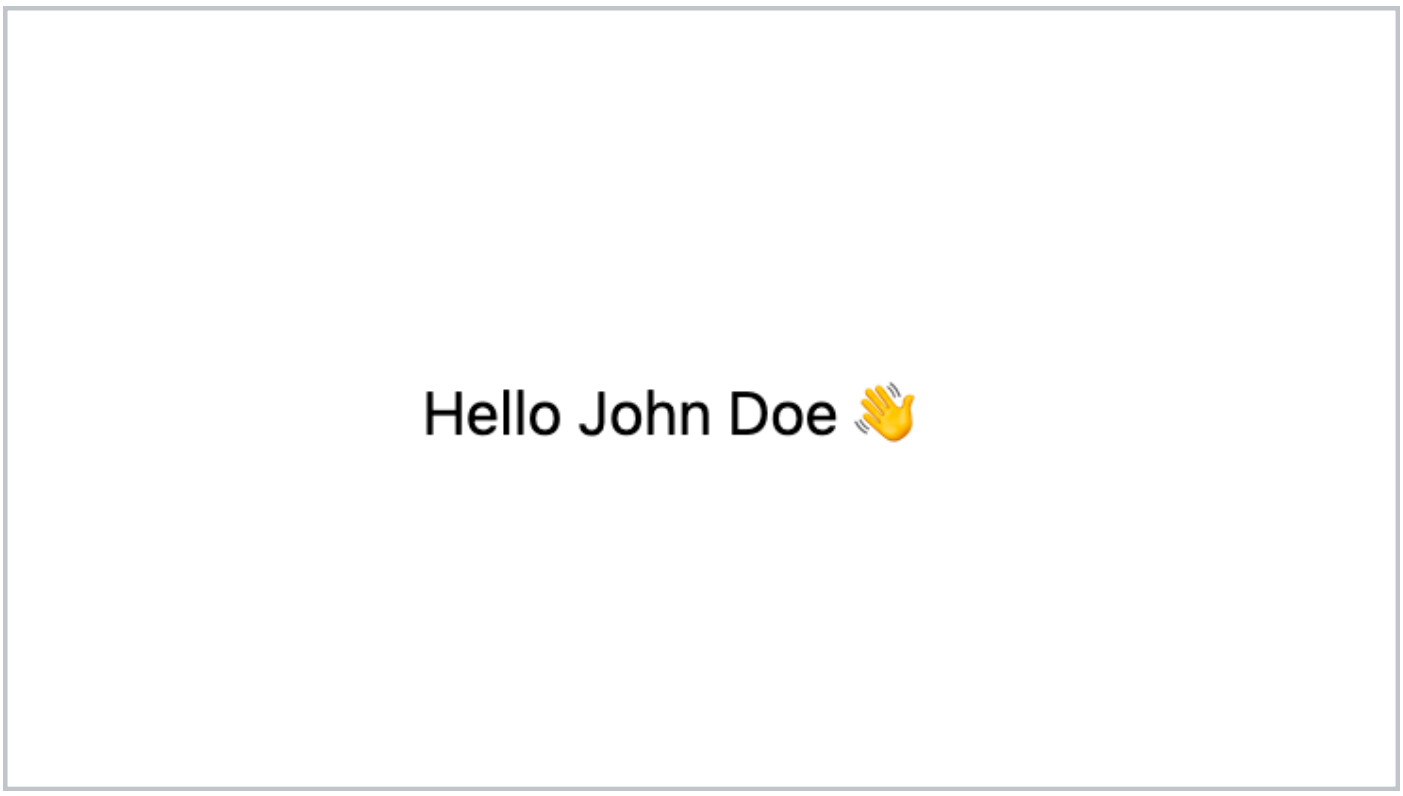
Monada to abstrakcyjny pojemnik na wartości, który udostępnia specjalne metody do przetwarzania tej wartości.

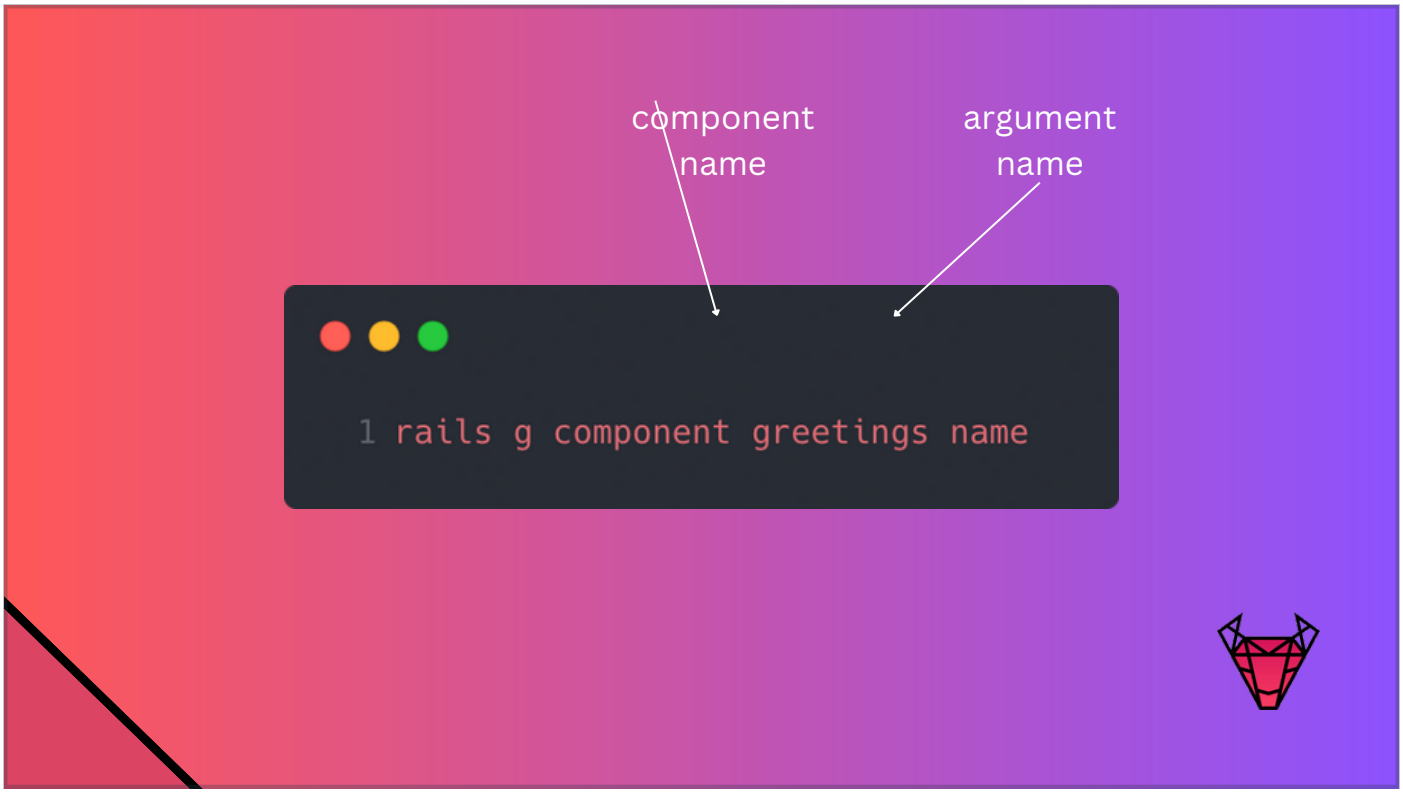
- Obiekty Ruby, które generują HTML
- Klasa Ruby + template
- Odizolowanie od pozostałej części aplikacji, testowalność, reużywalność

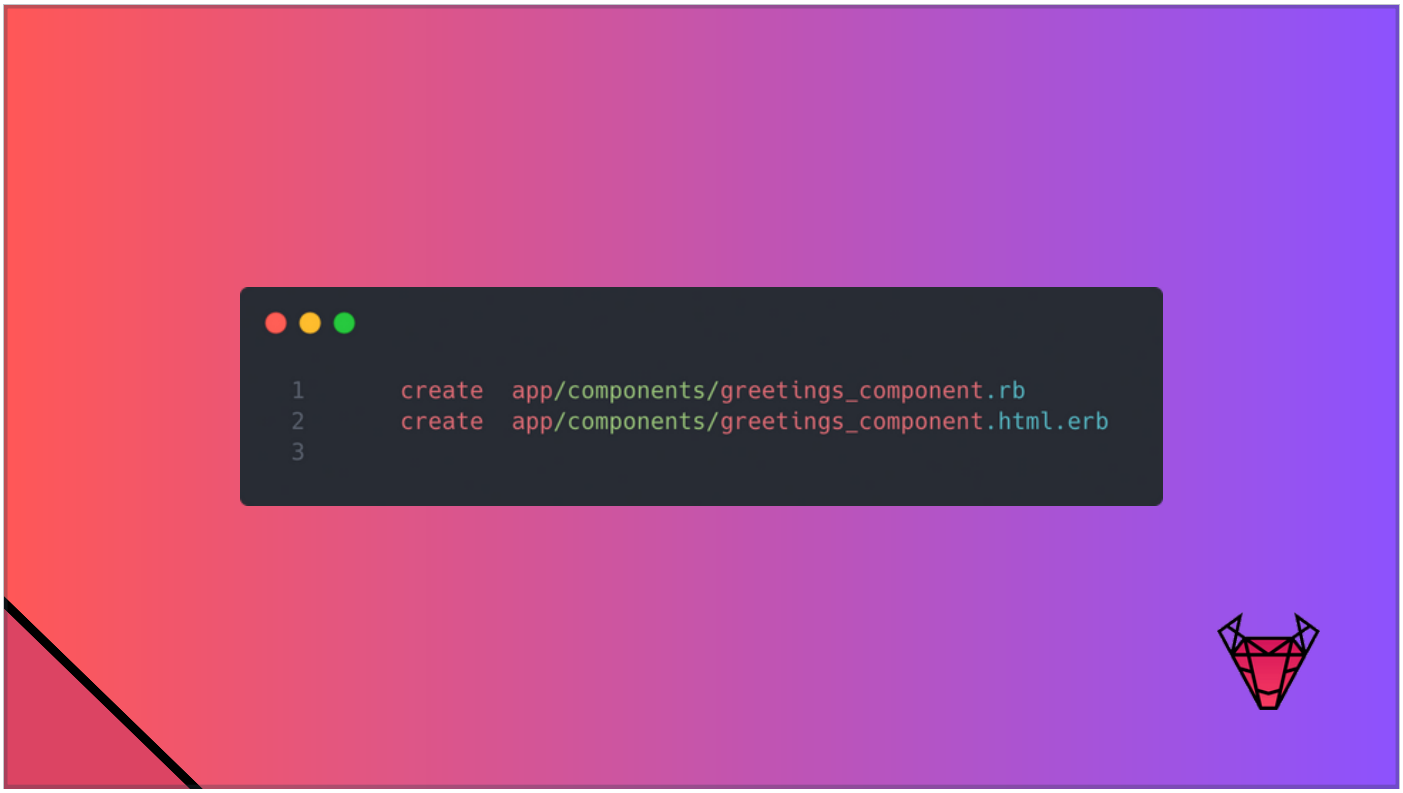
Monada to abstrakcyjny pojemnik na wartości, który udostępnia specjalne metody do przetwarzania tej wartości.


#app/views/greetings/index.html.erb

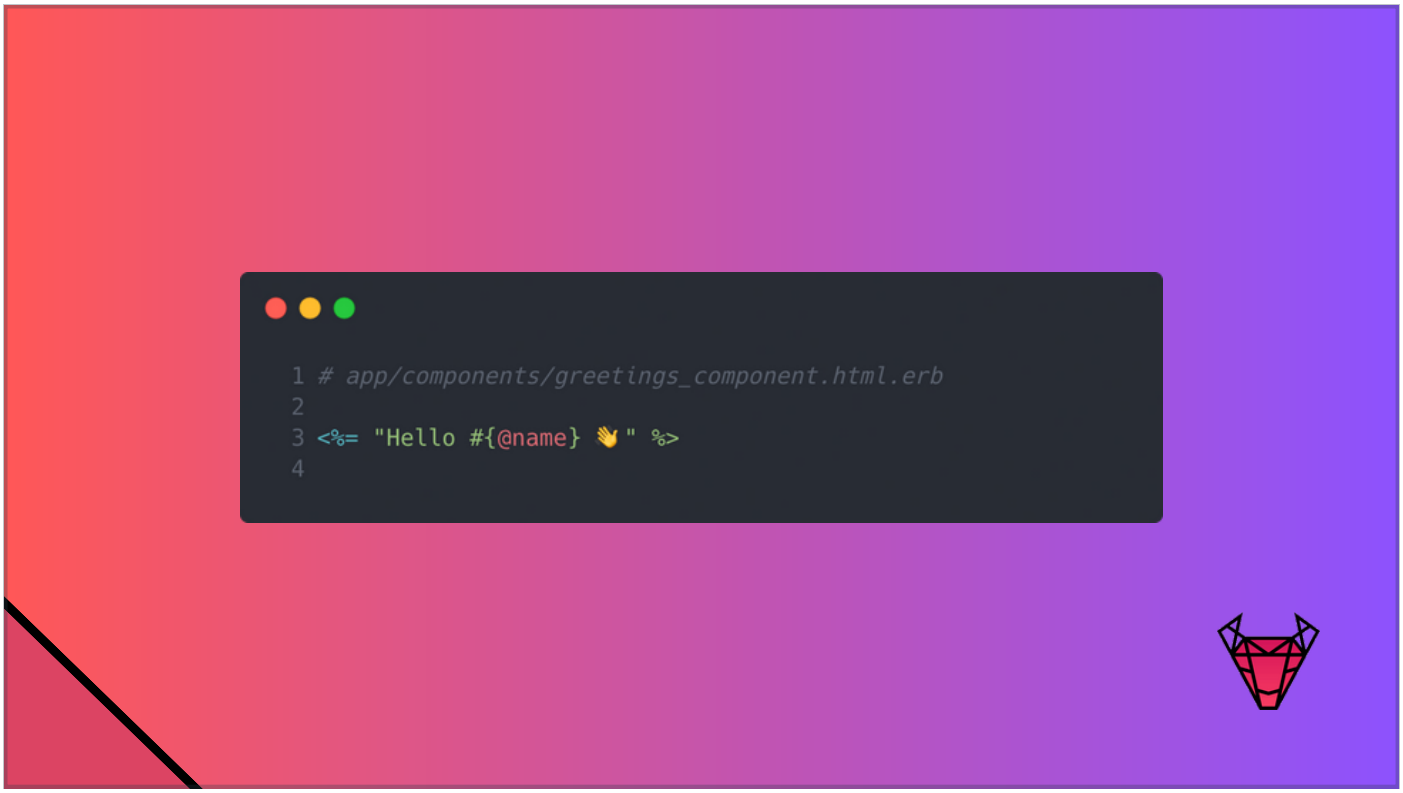

#app/views/greetings/index.html.erb
<%= "Hello #{current_user.name} 👋" %>


Hello John Doe 👋


component
name
argument
name
1 rails g component greetings name

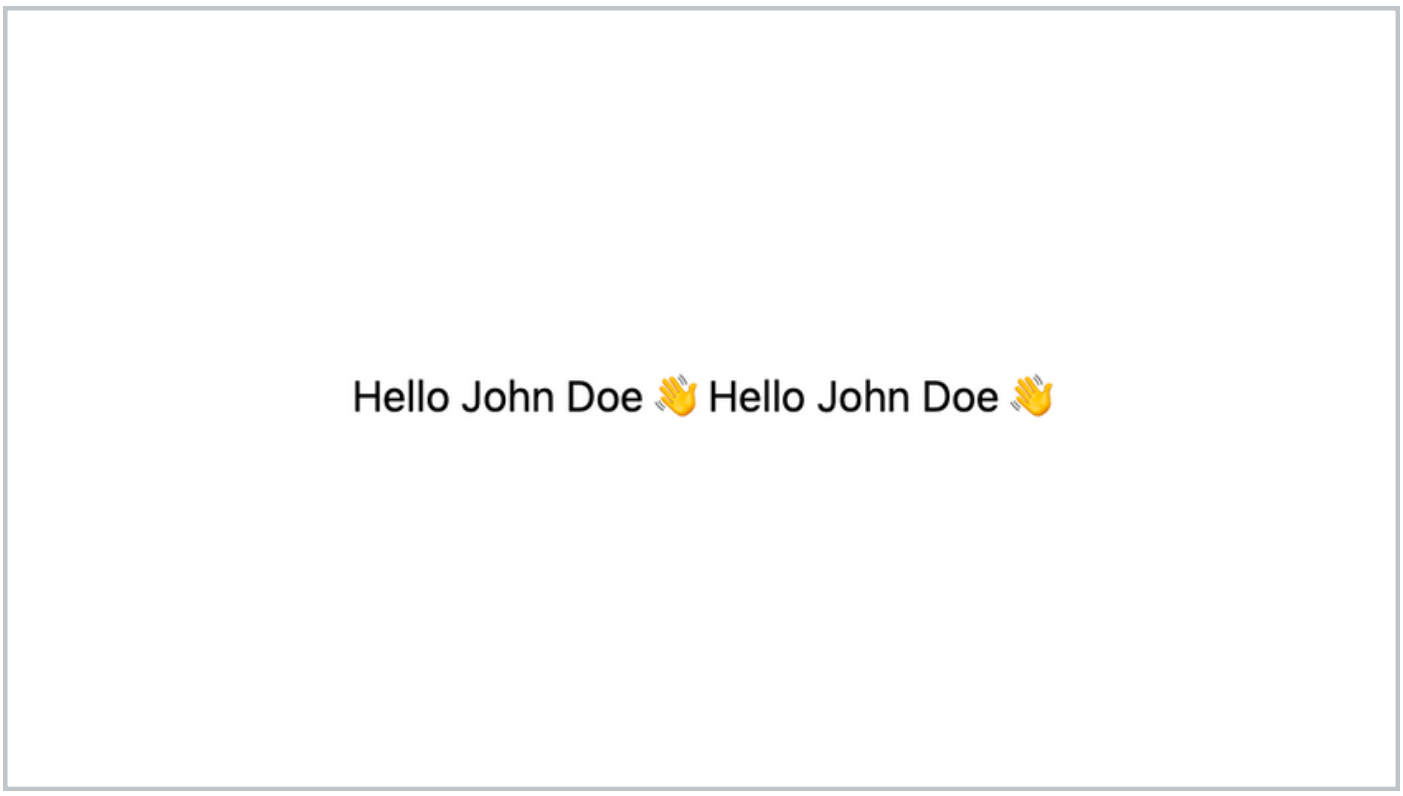

create app/components/greetings_component.rb
create app/components/greetings_component.html.erb


1 # app/components/greetings_component.rb
2
3 class GreetingsComponent < ViewComponent::Base
4   def initialize(name:)
5   @name = name
6   end
7 end
8

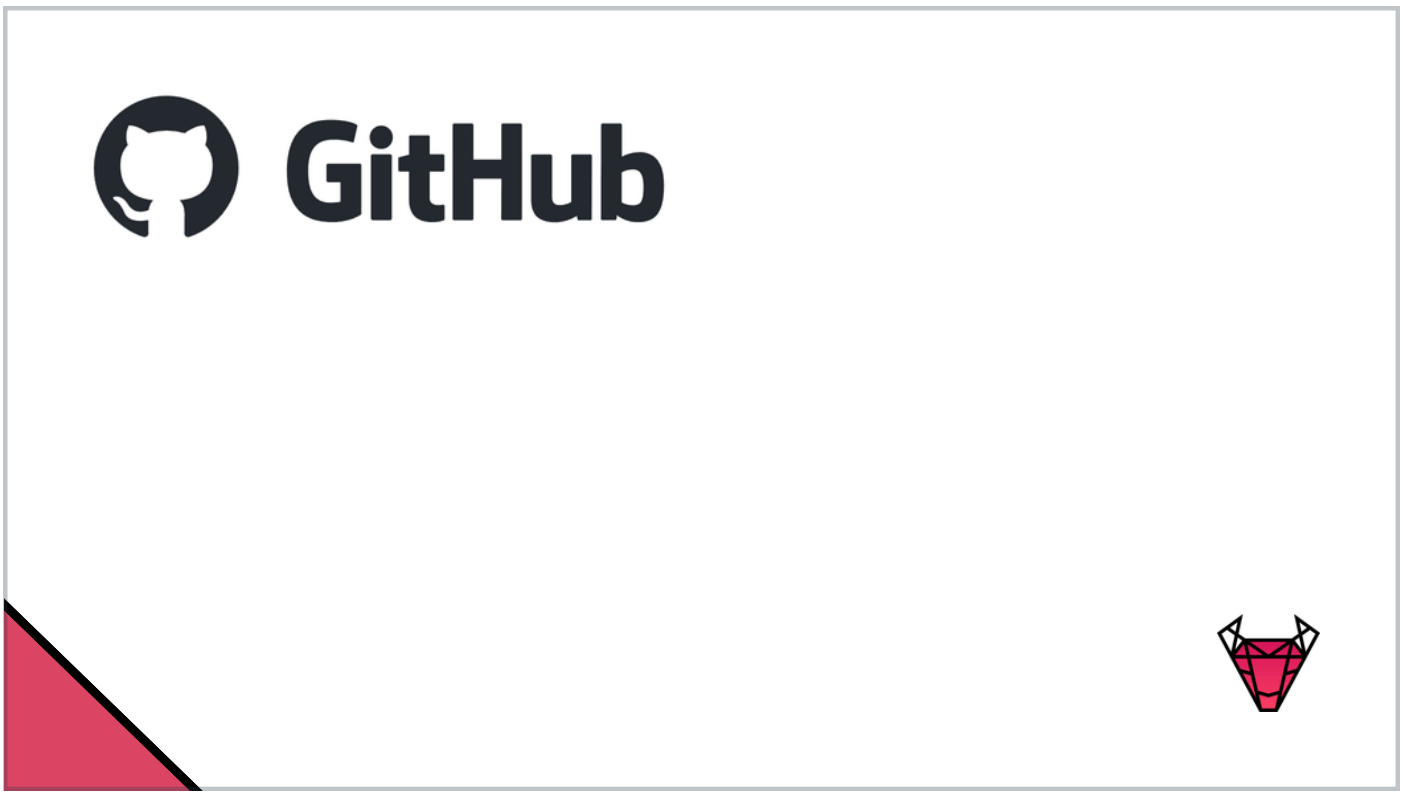


Monada to abstrakcyjny pojemnik na wartości, który udostępnia specjalne metody do przetwarzania tej wartości.

Monada to abstrakcyjny pojemnik na wartości, który udostępnia specjalne metody do przetwarzania tej wartości.

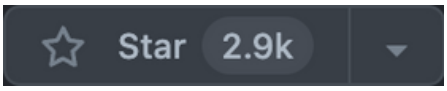

Hello John Doe 👋 Hello John Doe 👋

Kto i dlaczego?


GitHub

Aktualna stabilna wersja: **v2.82.0**

Prace trwają nad: **v3.0.0.rc5**



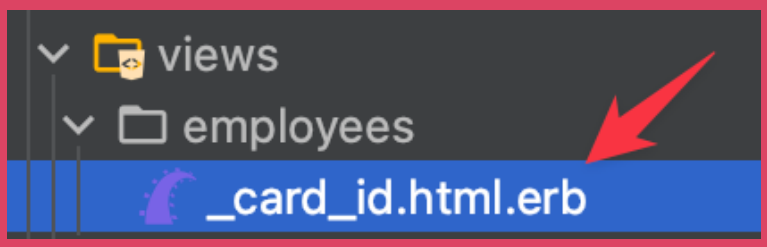
- łatwiej zarządzać rosnącą złożonością widoków

problem thousends copy pasting widoków

poprawiać jakość kodu np dla osób niepełnosprawnych
accessibility

piramida z testowaniem

Jaki jest problem z shared partials?



- ściśle powiązany kod

odnosi się do sytuacji gdzie partial je wykorzystywany w dwóch widokach. Zmiana w partialu powoduje zmianę widoku w tych dwóch stronach. To utrudnia modyfikowanie, rozszerzanie lub utrzymanie kodu.

- ściśle powiązany kod
- ukryte argumenty

patrzymy na render partiala i nie wiemy jakich danych używa(no chyba, że używamy locals do przekazywania danych)

- ściśle powiązany kod
- ukryte argumenty
- nieprzewidywalny przepływ danych

nie wiadomo gdzie zmienne instancyjne zostały zainicializowane. Mogą byc przecież zainicjalizowane w beforach :X

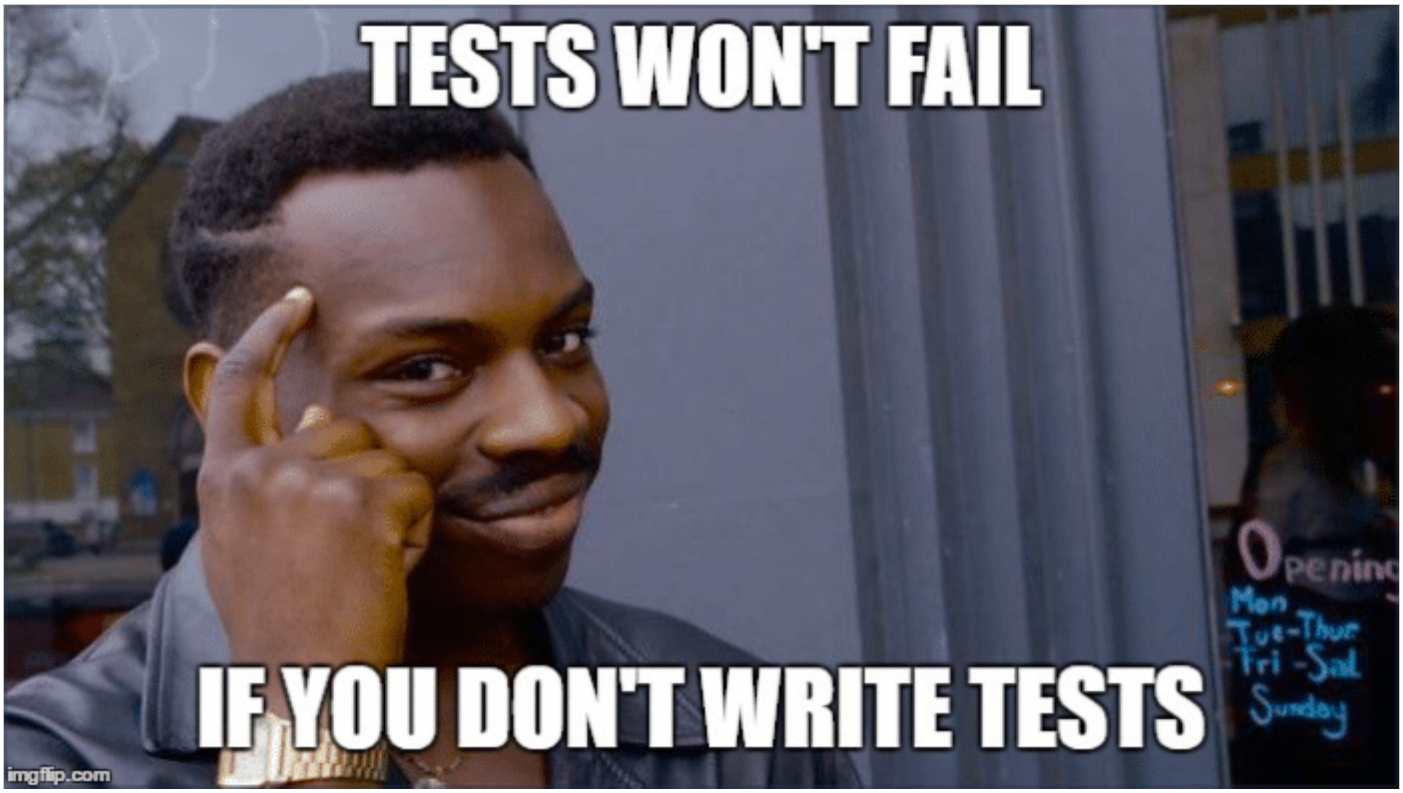
- ściśle powiązany kod
- ukryte argumenty
- nieprzewidywalny przepływ danych
- używanie helperów

- brak enkapsulacji (wszystkie metody są jawne i dostępne z poziomu widoków)

- zanieczyszczenie globalnej przestrzeni nazw. Konflikt nazw

- ściśle powiązany kod
- ukryte argumenty
- nieprzewidywalny przepływ danych
- używanie helperów
- trudności z testowaniem?

- brak enkapsulacji (wszystkie metody są jawne i dostępne z poziomu widoków)


TESTS WON'T FAIL
IF YOU DON'T WRITE TESTS
imgflip.com

Module: RSpec::Rails::ViewExampleGroup::ExampleMethods

Defined in: lib/rspec/rails/example/view_example_group.rb

Overview

DSL exposed to view specs.

Instance Method Summary

(expand)

params render ~~response~~ stub_template ~~template~~ view

Instance Method Details

- (Object) params

Provides access to the params hash that will be available within the view.

```
params[:foo] = 'bar'
```

[]

- (Object) render
- (Object) render({:partial = > path_to_file})
- (Object) render({:partial = > path_to_file}, {... locals ...})
- (Object) render({:partial = > path_to_file}, {... locals ...})

Delegates to ActionView::Base#render, so see documentation on that for more info.

The only addition is that you can call render with no arguments, and RSpec will pass the top level description to render:

```
describe "widgets/new.html.erb" do
  it "shows all the widgets" do
    render # => view.render(:file => "widgets/new.html.erb")
    # ...
  end
end
```

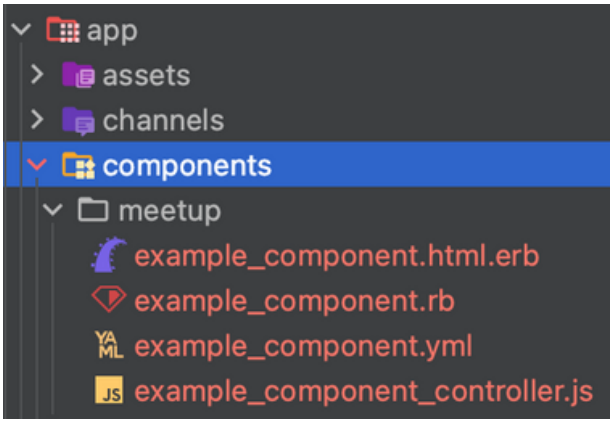
[]


1 # views/greetings/hello_spec.rb
2
3 require 'rails_helper'
4
5 describe 'hello partial' do
6 it 'succeeds' do
7 render partial: 'greetings/hello', locals: { name: 'John Doe' }
8
9 expect(rendered).to have_content 'Hello John Doe 👋'
10 end
11 end

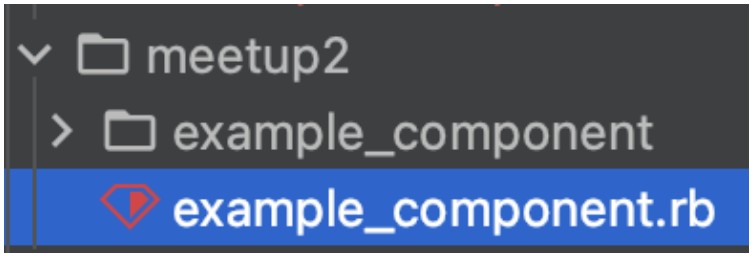
1. Reużywalność kodu
2. Dane przekazywane w jawny sposób
3. Nie korzystamy z helperów, bo mamy do każdego templata klasę ruby

Generatory / Konfiguracja

```
1 rails g component meetup/example title --stimulus --locale
```

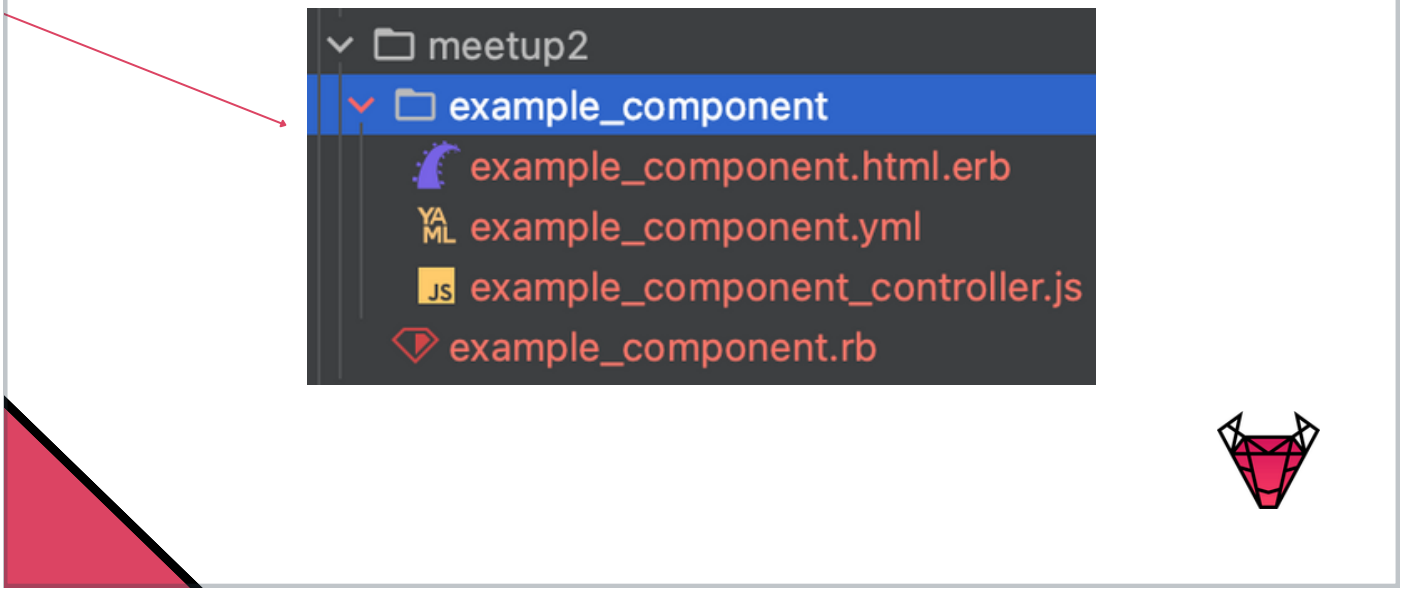


```
1 rails g component meetup2/example title --stimulus --locale --sidecar
```




meetup2
example_component
example_component.html.erb
example_component.yml
example_component_controller.js
example_component.rb

locals



domyślnie te tłumaczenia są izolowane do jednego komponentu. Z jednej strony to jest fajne, ale z drugiej strony zarządanie wieloma plikami yml może być kłopotliwe.

I18n integration (alternative)

palkan/view_component-contrib

ViewComponent recently added (experimental) I18n support, which allows you to have **isolated** localization files for each component. Isolation rocks, but managing dozens of YML files spread accross the project could be tricky, especially, if you rely on some external localization tool which creates these YMLs for you.

We provide an alternative (and more *classic*) way of dealing with translations—**namespacing**. Following the convention over configuration, put translations under `<locale>.view_components.<component_scope>` key, for example:

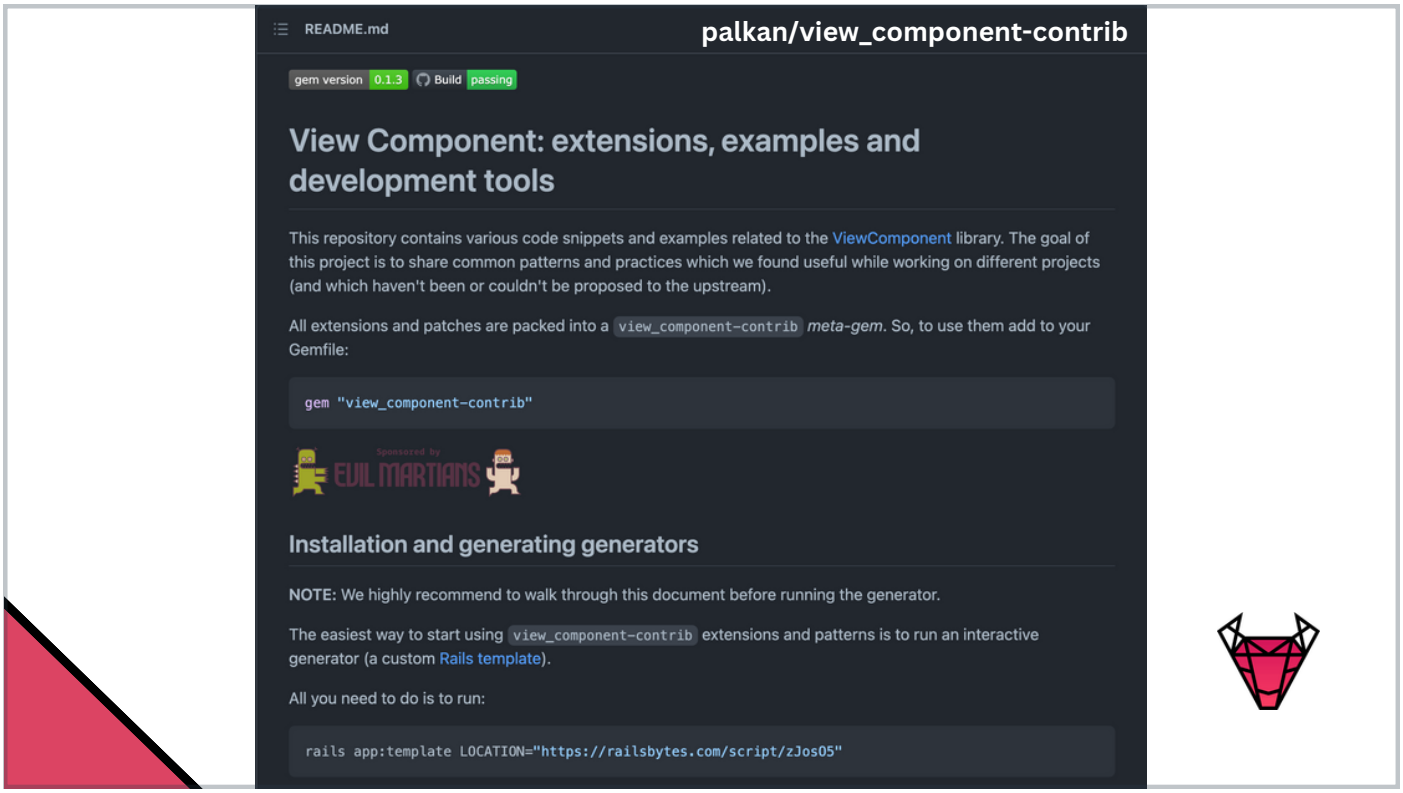
```yaml
en:
  view_components:
    login_form:
      submit: "Log in"
    nav:
      user_info:
        login: "Log in"
        logout: "Log out"
```

And then in your components:

```erb
<!-- login_form/component.html.erb -->
<button type="submit"><%= t(".submit") %></button>

<!-- nav/user_info/component.html.erb -->
<a href="/logout"><%= t(".logout") %></a>
```

If you're using `ViewComponentContrib::Base`, you already have translation support included. Othwerwise you must include the module yourself:

```ruby
class ApplicationViewComponent < ViewComponent::Base
  include ViewComponentContrib::TranslationHelper
end
```


README.md
palkan/view_component-contrib
gem version 0.1.3
Build passing
View Component: extensions, examples and development tools
This repository contains various code snippets and examples related to the ViewComponent library. The goal of this project is to share common patterns and practices which we found useful while working on different projects (and which haven't been or couldn't be proposed to the upstream).
All extensions and patches are packed into a view_component-contrib meta-gem. So, to use them add to your Gemfile:
gem "view_component-contrib"
Sponsored by
EVIL MARTIANS
Installation and generating generators
NOTE: We highly recommend to walk through this document before running the generator.
The easiest way to start using view_component-contrib extensions and patterns is to run an interactive generator (a custom Rails template).
All you need to do is to run:
rails app:template LOCATION="https://railsbytes.com/script/zJosO5"

Inline rendering

```
1 rails g component meetup3/example title --inline
```

```
1       create  app/components/meetup3/example_component.rb
```

```ruby
# app/components/meetup3/example_component.rb

module Meetup3
  class ExampleComponent < ViewComponent::Base
    def initialize(title:)
      @title = title
    end

    def call
      content_tag :h1, 'Hello world!'
    end
  end
end

```

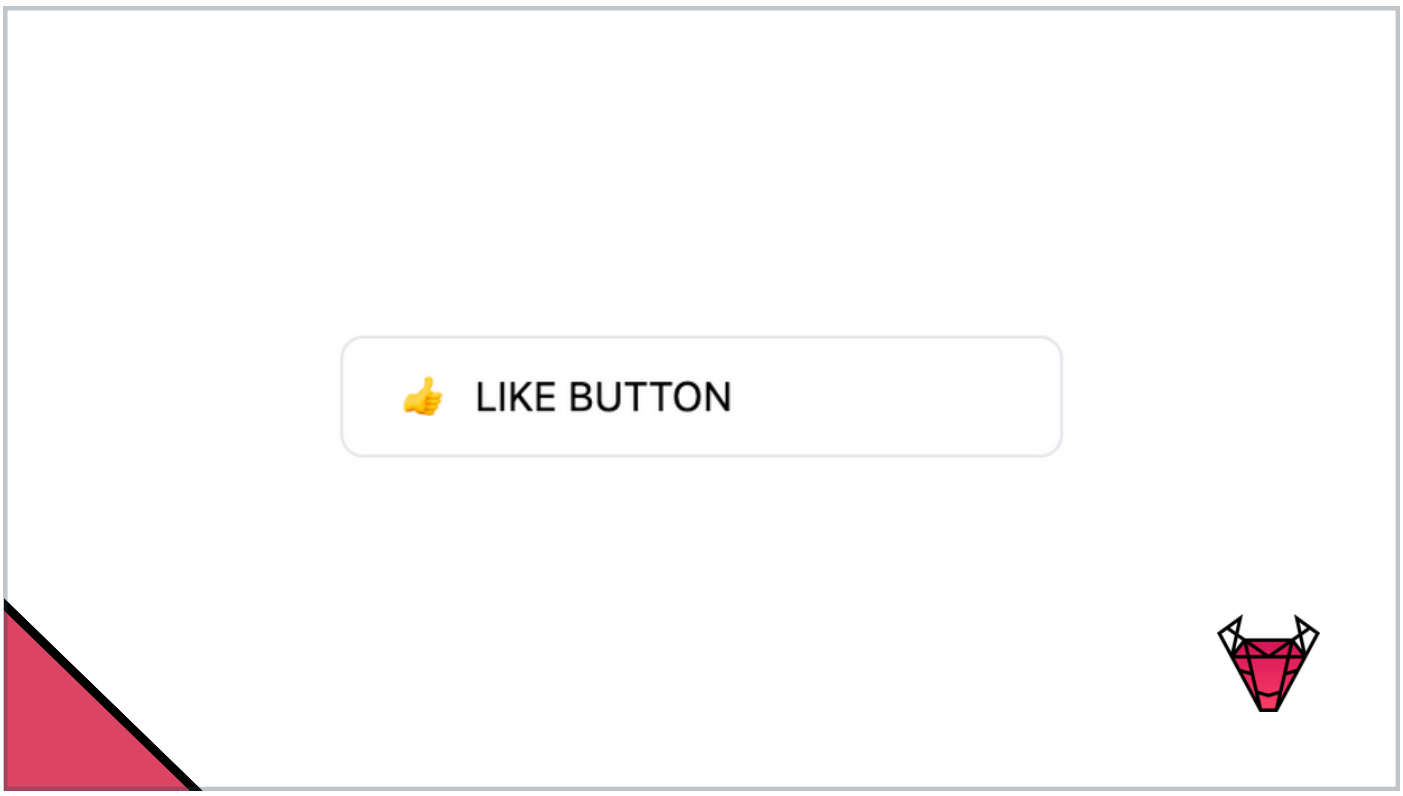

Default
Dark
Red
Green
Yellow
Indigo
Purple
Pink

```ruby
class InlineComponent < ViewComponent::Base
  def call
    if active?
      link_to "Cancel integration", integration_path, method: :delete
    else
      link_to "Integrate now!", integration_path
    end
  end
end

```

Parent class

```ruby
module Shared
  class ButtonComponent < ViewComponent::Base
    attr_reader :label, :icon

    def initialize(label:, icon:)
      @label = label
      @icon = icon
    end
  end
end
```

```erb
<button type="button" class="border rounded-lg">
  <% if icon %> <i class="mr-2"><%= icon %></i> <% end %>
  <%= label %>
</button>

```


👍 LIKE BUTTON

```
1 rails g component shared/like_button2 label --parent Shared::ButtonComponent
```

```ruby
module Shared
  class LikeButton2Component < Shared::ButtonComponent
    def initialize(label:)
      super(label:, icon: '👍')
    end
  end
end
```

```erb
1 <%= render Shared::LikeButton2Component.new(label: 'LIKE BUTTON NO INLINE') %>
```

```erb
<div class="bg-amber-500 w-56 rounded-md ">
  <% render_parent %>
</div>
```

dry-initialize

```ruby
class BaseViewComponent < ViewComponent::Base
  extend Dry::Initializer
end
```

```ruby
# config/initializers/view_component.rb

Rails.application.configure do
  config.view_component.component_parent_class = 'BaseViewComponent'
end
```

```ruby
module Meetup4
  class ExampleComponent < BaseViewComponent
    # def initialize(title:,
    #   @title = title
    # end
    option :title
  end
end

```

Przekazywanie danych

Content

Content

```erb
<%= render Meetup5::ButtonComponent.new do %>
  BUTTON CONTENT
<% end %>
```

```erb
#app/components/meetup5/button_component.html.erb
<div class="border rounded-md w-fit bg-amber-500 p-4">
  <%= content %>
</div>
```

BUTTON CONTENT

Content

```erb
<%= render Meetup5::ButtonComponent.new do %>
  BUTTON CONTENT
<% end %>
```

```erb
# app/views/greetings/index.html.erb
<%= render Meetup5::ButtonComponent.new.with_content('Button with content2') %>
```

Button with content2

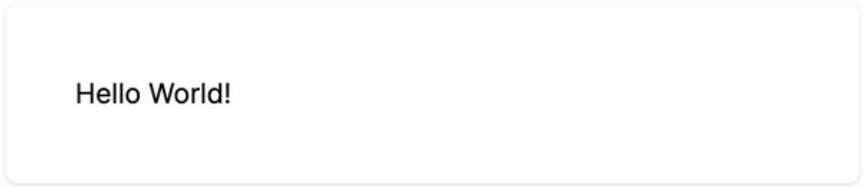
Content

```ruby
module Shared
  class ContainerComponent < ViewComponent::Base
  end
end
```

```erb
# app/components/shared/container_component.html.erb
<div class='bg-white p-10 flex flex-col mb-5 rounded-md shadow'>
  <%= content %>
</div>

```

Content

```ruby
render(Shared::ContainerComponent.new.with_content('Hello World!'))
```



Slots

```ruby
# app/components/shared/container_component.rb:8
module Shared
  class ContainerComponent < ViewComponent::Base
    renders_one :header
  end
end
```

```erb
app/components/shared/container_component.html.erb
<div class='bg-white p-10 flex flex-col mb-5 rounded-md shadow'>
  <%= content_tag :h3, header, class: 'text-xl font-bold mb-4' if header %>

  <%= content %>
</div>
```

Slots

```ruby
render(Shared::ContainerComponent.new) do |component|
  component.header { 'Header' }
  "Content"
end
```

Header

Content

Slots#2

```erb
<%= render Shared::ContainerComponent.new do |container| %>
  <% container.with_header { 'Header' } %>
  <%= render Shared::Show::ObjectDetailsComponent.new do |object_details| %>
    <% object_details.with_detail(label: 'Detail 1').with_content('Content 1') %>
    <% object_details.with_detail(label: 'Detail 2').with_content('Content 2') %>
    <% object_details.with_detail(label: 'Detail 3').with_content('Content 3') %>
    <% object_details.with_detail do %>
      <%= link_to 'Delete book', 'Edit object', class: 'text-red-500' %>
    <% end %>
  <% end %>
<% end %>

```

Slots#2

```erb
<%= render Shared::ContainerComponent.new do |container| %>
  <% container.with_header { 'Header' } %>
  <%= render Shared::Show::ObjectDetailsComponent.new do |object_details| %>
    <% object_details.with_detail(label: 'Detail 1').with_content('Content 1') %>
    <% object_details.with_detail(label: 'Detail 2').with_content('Content 2') %>
    <% object_details.with_detail(label: 'Detail 3').with_content('Content 3') %>
    <% object_details.with_detail do %>
      <%= link_to 'Delete book', 'Edit object', class: 'text-red-500' %>
    <% end %>
  <% end %>
<% end %>

```

Slots#2

```ruby
module Shared
  module Show
    class ObjectDetailsComponent < ViewComponent::Base
      renders_many :details, Shared::Show::ObjectDetailComponent
    end
  end
end

```

Slots#2

```
<%= render Shared::ContainerComponent.new do |container| %>
  <% container.with_header { 'Header' } %>
  <%= render Shared::Show::ObjectDetailsComponent.new do |object_details| %>
    <% object_details.with_detail(label: 'Detail 1').with_content('Content 1') %>
    <% object_details.with_detail(label: 'Detail 2').with_content('Content 2') %>
    <% object_details.with_detail(label: 'Detail 3').with_content('Content 3') %>
    <% object_details.with_detail do %>
      <%= link_to 'Delete book', 'Edit object', class: 'text-red-500' %>
    <% end %>
  <% end %>
<% end %>

```

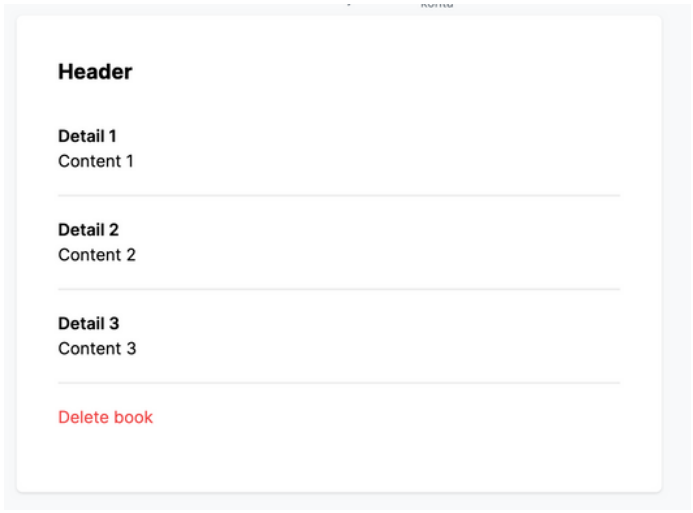
Slots#2

```ruby
# frozen_string_literal: true

module Shared
  module Show
    class ObjectDetailComponent < ViewComponent::Base
      attr_reader :label
      def initialize(label: nil)
        @label = label
      end
    end
  end
end

```

Slots#2

```erb
# app/components/shared/show/object_detail_component.html.erb
<div class='border-b-2 border-gray_light flex flex-col p-5 pl-0 last:!border-b-0'>
  <div class='font-semibold'>
    <%= label %>
  </div>
  <%= content %>
</div>
```

Slots#2


Header
Detail 1
Content 1
Detail 2
Content 2
Detail 3
Content 3
Delete book

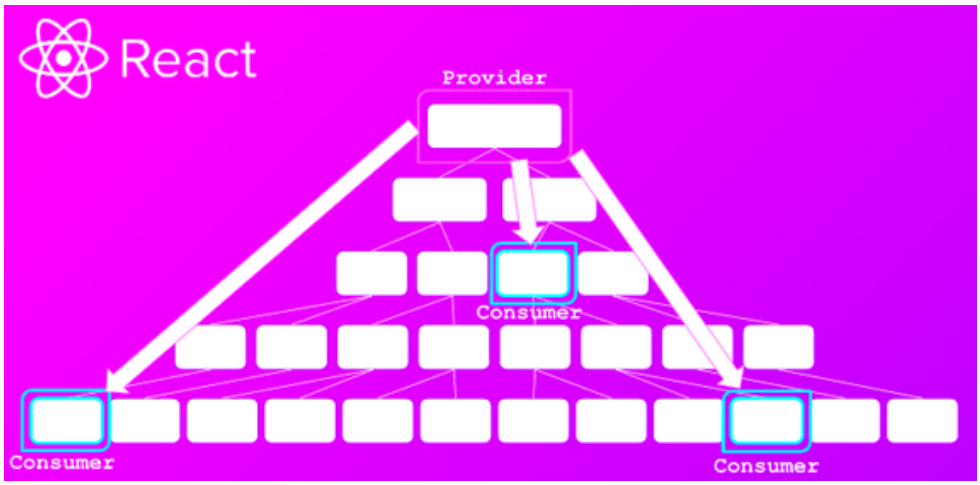
Slots#2

```erb
<%= render Shared::ContainerComponent.new do |container| %>
  <% container.with_header { 'Header' } %>
  <%= render Shared::Show::ObjectDetailsComponent.new do |object_details| %>
    <% object_details.with_detail(label: 'Detail 1').with_content('Content 1') %>
    <% object_details.with_detail(label: 'Detail 2').with_content('Content 2') %>
    <% object_details.with_detail(label: 'Detail 3').with_content('Content 3') %>
    <% object_details.with_detail do %>
      <%= link_to 'Delete book', 'Edit object', class: 'text-red-500' %>
    <% end %>
  <% end %>
<% end %>

```

Slots#2

```erb
<div class='bg-white p-10 flex flex-col mb-5 rounded-md shadow'>
  <%= content_tag :h3, 'Header', class: 'text-xl font-bold mb-4' %>

  <div class='border-b-2 border-gray_light flex flex-col p-5 pl-0 last:!border-b-0'>
    <div class='font-semibold'>
      <%= 'Detail 1' %>
    </div>
    <%= 'Content 1' %>
  </div>

  <div class='border-b-2 border-gray_light flex flex-col p-5 pl-0 last:!border-b-0'>
    <div class='font-semibold'>
      <%= 'Detail 2' %>
    </div>
    <%= 'Content 2' %>
  </div>

  <div class='border-b-2 border-gray_light flex flex-col p-5 pl-0 last:!border-b-0'>
    <div class='font-semibold'>
      <%= 'Detail 3' %>
    </div>
    <%= 'Content 3' %>
  </div>

  <div class='border-b-2 border-gray_light flex flex-col p-5 pl-0 last:!border-b-0'>
    <%= link_to 'Delete book', 'Edit object', class: 'text-red-500' %>
  </div>
</div>

```

Inne


React
Provider
Consumer
Consumer
Consumer

Renderowanie komponentów z controllera

```ruby
# app/controllers/home_controller.rb
def show
  render(ExampleComponent.new(title: "My Title"))
end
```

dry-effects

ApplicationController

```ruby
include Dry::Effects::Handler.Reader(:current_user)

around_action :set_current_user

private

def set_current_user
  # Assuming you have `#current_user` method defined:
  with_current_user(current_user) { yield }
end
```

BaseViewComponent

```ruby
include Dry::Effects.Reader(:current_user, default: nil)
```

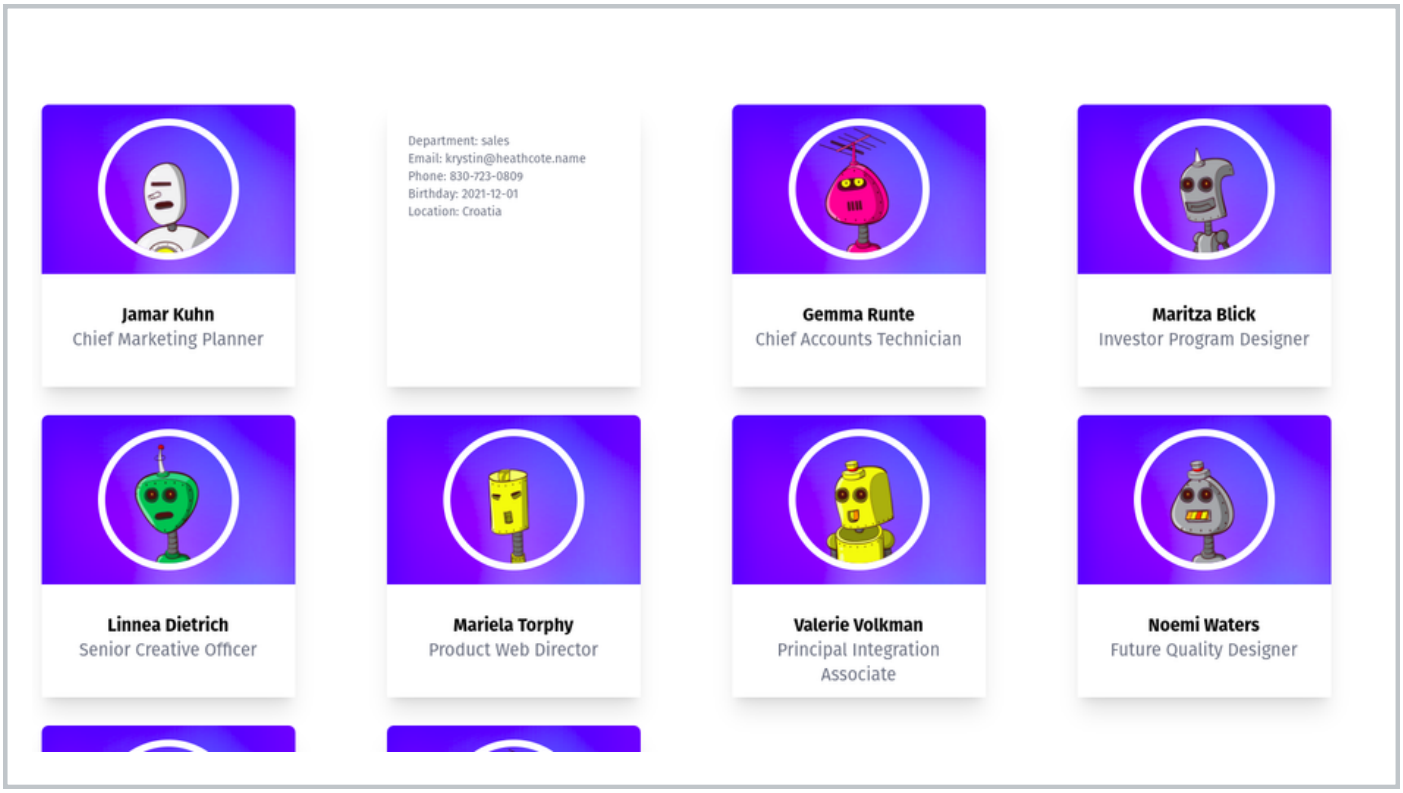
conditional rendering

```ruby
class ExampleComponent < ViewComponent::Base
  def initialize(user:)
    @user = user
  end

  def render?
    @user.requires_confirmation?
  end
end
```

collections

```ruby
class ProductComponent < ViewComponent::Base
  def initialize(product:)
    @product = product
  end
end
```

```erb
<div class="flex gap-2">
    <%= render(ProductComponent.with_collection(@products, notice: "hi")) %>
</div>
```



Monada to abstrakcyjny pojemnik na wartości, który udostępnia specjalne metody do przetwarzania tej wartości.

Previews

previews

```
1  rails g component Shared::ImageModalComponent --preview
```

previews

```
1  rails g component Shared::ImageModalComponent --preview
```

test/components/previews/shared/image_modal_component_preview.rb

previews

http://0.0.0.0:3000/rails/view_components

Publishers/Promotions/Status Component

- active
- expired
- on_hold
- waiting

Shared/Image Modal Component

- default
- sm_size
- with_custom_url

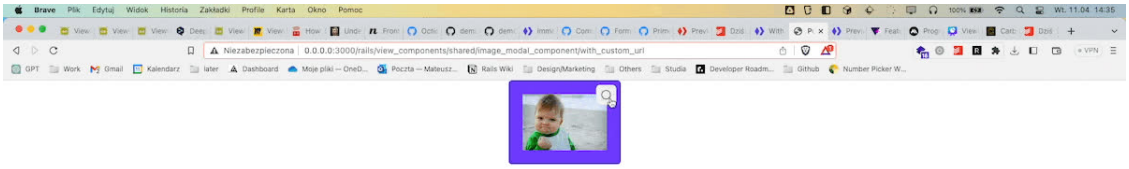
Shared/Container Component

- default

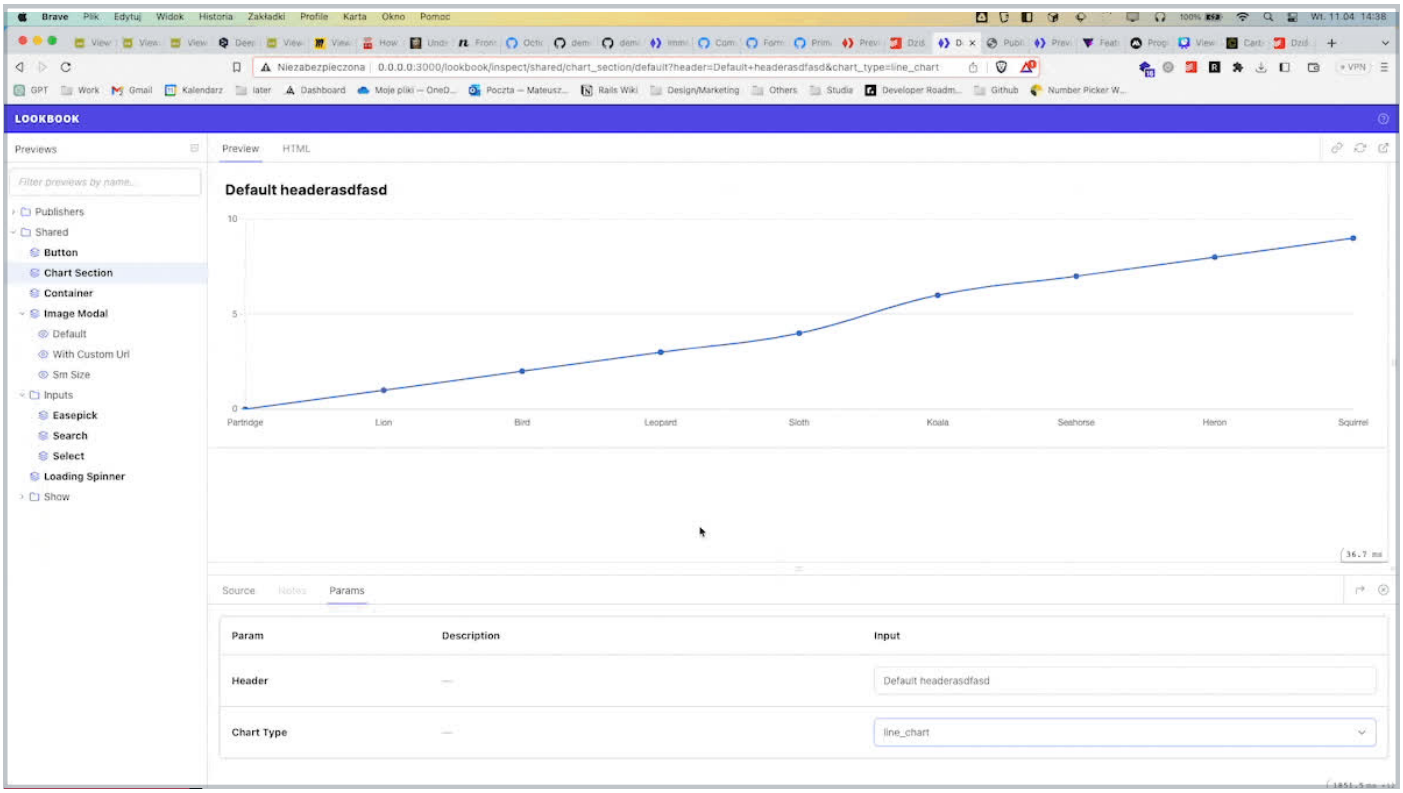
Shared/Button Component

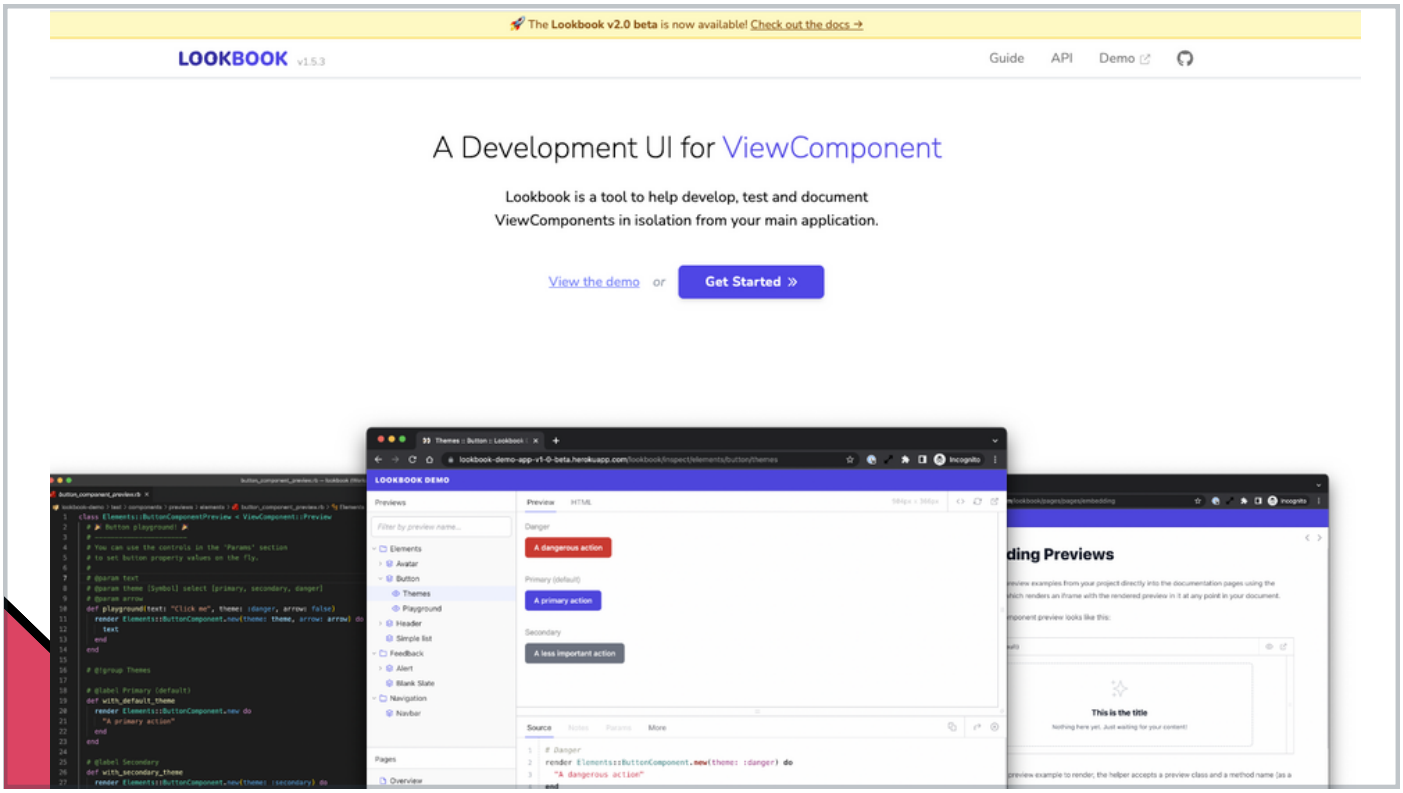
- default

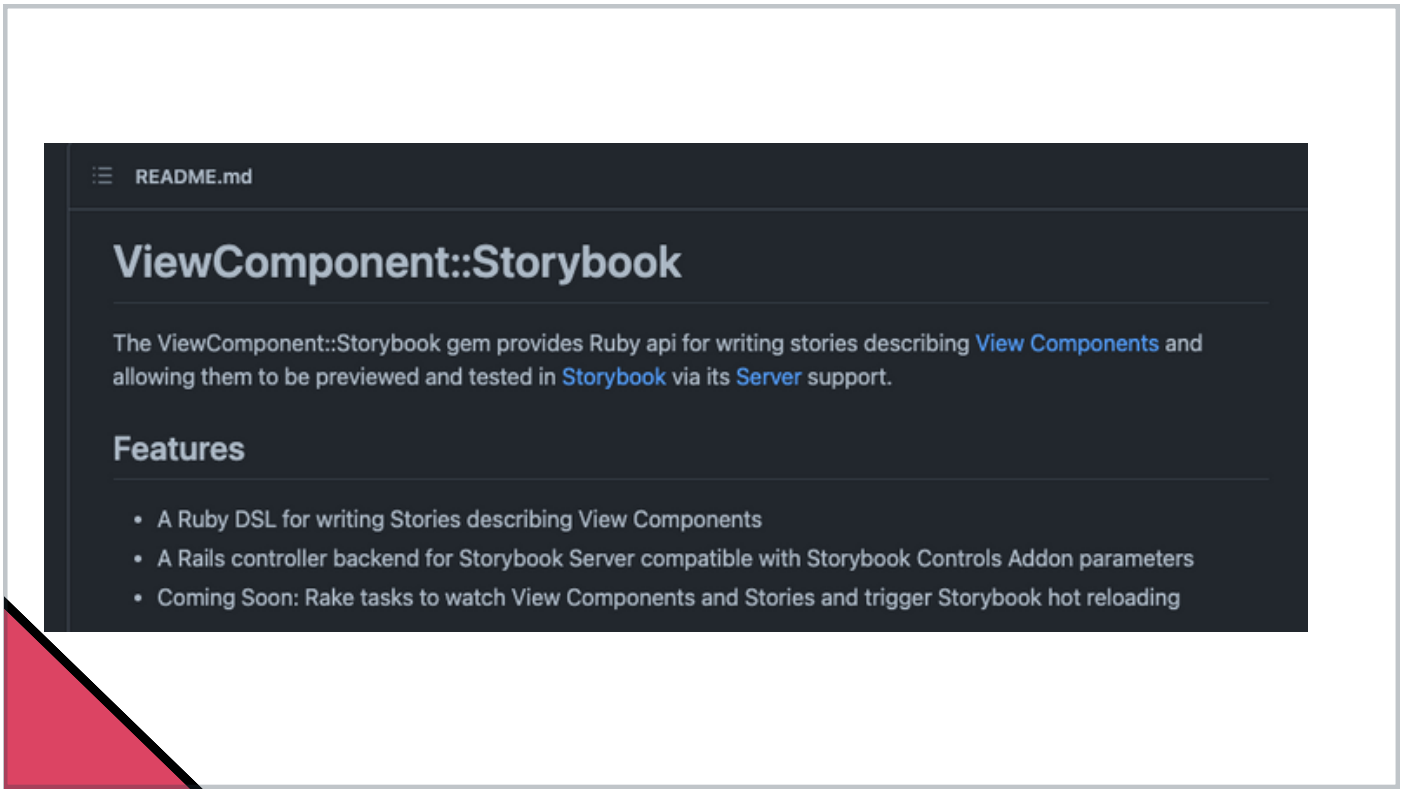
```ruby
# components/previews/shared/image_modal_component_preview.rb

module Shared
  class ImageModalComponentPreview < ViewComponent::Preview
    layout 'component_preview'

    def default
      render(Shared::ImageModalComponent.new(image_url:
PublicationOffer.first.image_url))
    end

    def with_custom_url
      render(Shared::ImageModalComponent.new(image_url:
'https://images.theconversation.com/files/38926/original/5cwx89t4-
1389586191.jpg?ixlib=rb-1.1.0&q=45&auto=format&w=926&fit=clip'))
    end

    def sm_size
      render(Shared::ImageModalComponent.new(image_url:
PublicationOffer.first.image_url, size: 'sm'))
    end
  end
end

```

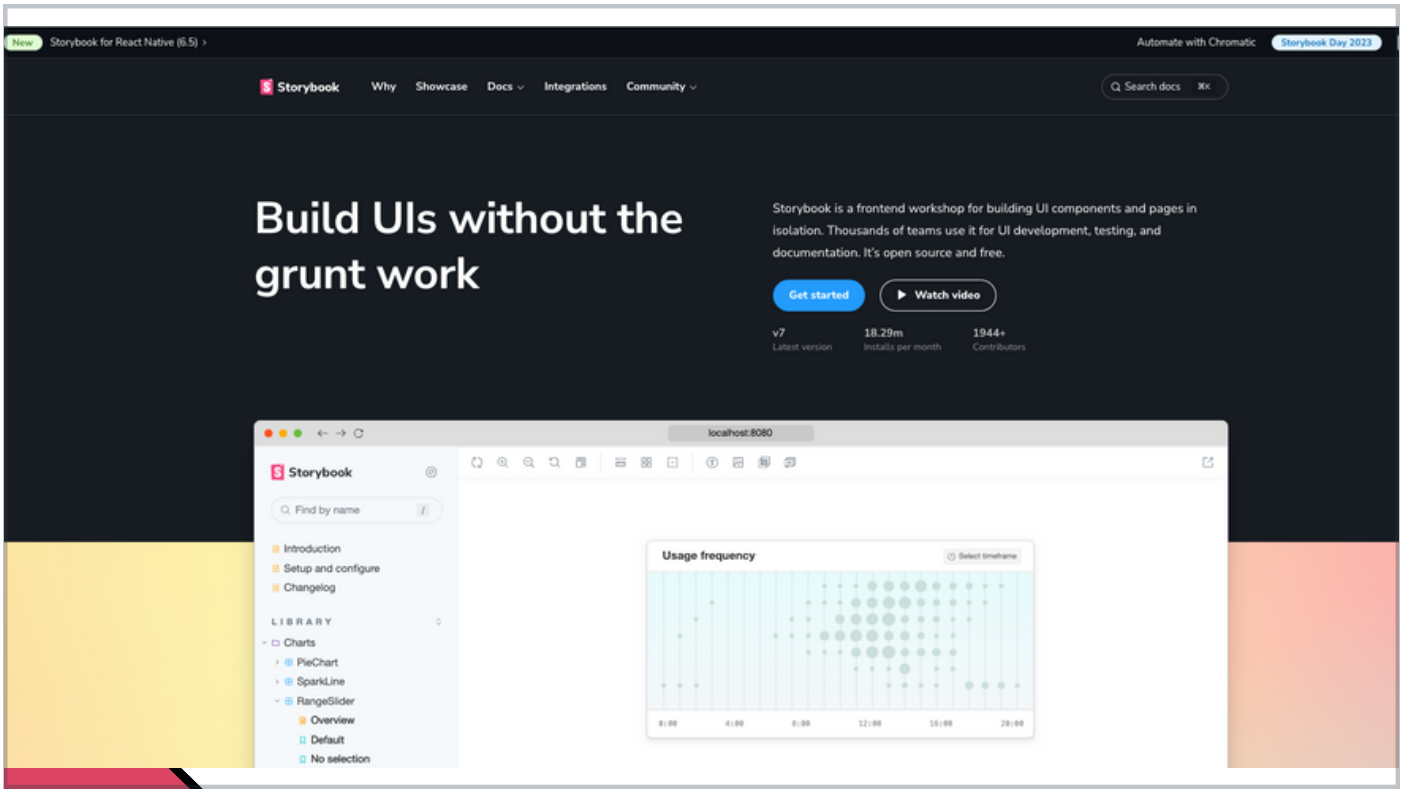

Brave Plik Edytuj Widok Historia Zakładki Profile Karta Okno Pomoc
Niezabezpieczona
0.0.0.0:3000/rails/view_components/shared/image_modal_component/with_custom_url

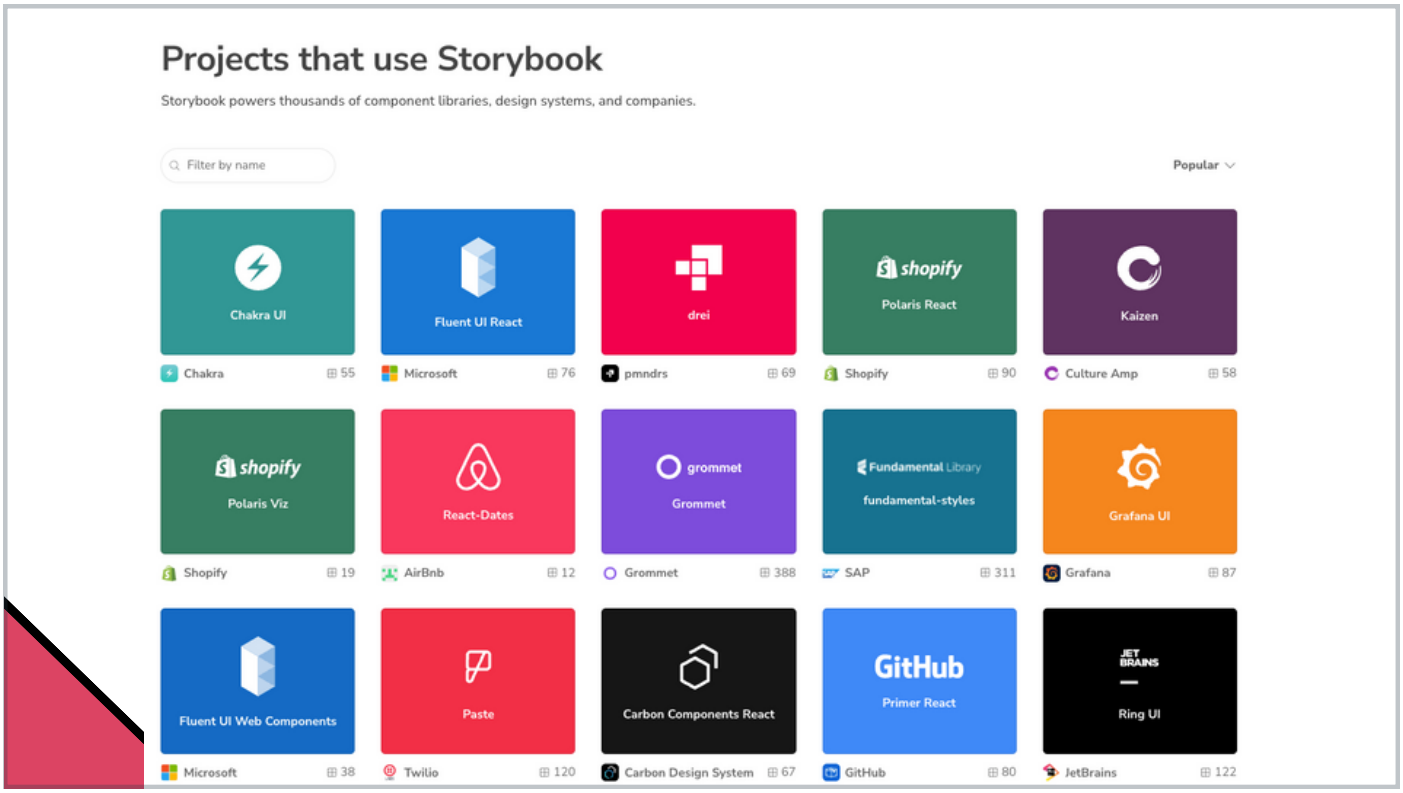
```ruby
# @param header [String]
# @param chart_type select { choices: [bar_chart, line_chart] }
def default(header: 'Default header', chart_type: 'line_chart')
  chart_data = Array.new(10) do |i|
    [FFaker::AnimalUS.common_name, i]
  end
  render(Shared::ChartSectionComponent.new(chart_data:, header:, chart_type:))
end
```


LOOKBOOK
Previews
Filter previews by name...
Publishers
Shared
Button
Chart Section
Container
Image Modal
Default
With Custom Url
Sm Size
Inputs
Easepick
Search
Select
Loading Spinner
Show
Preview
HTML
Default headerasdfasd
10
5
0
Partridge
Lion
Bird
Leopard
Sloth
Koala
Seahorse
Heron
Squirrel
Source
Notes
Params
Param
Description
Input
Header
Default headerasdfasd
Chart Type
line_chart

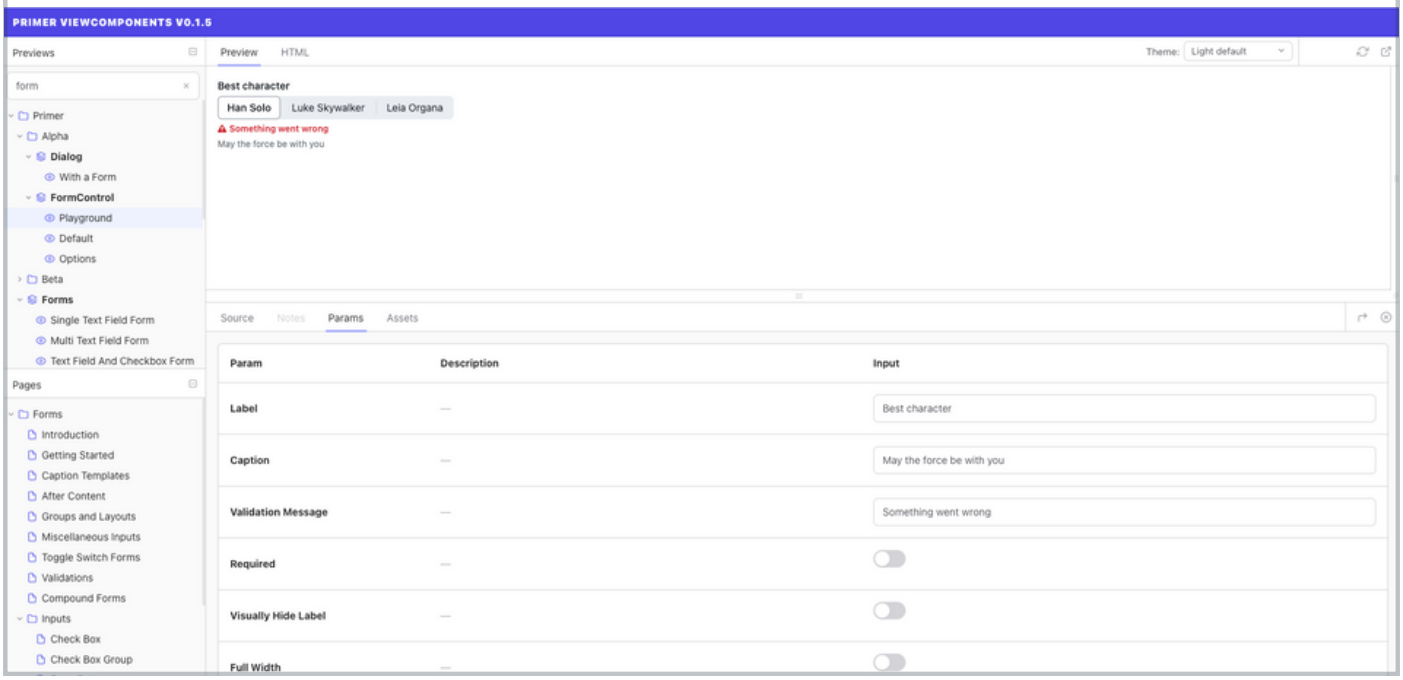

The Lookbook v2.0 beta is now available! Check out the docs →
LOOKBOOK v1.5.3
Guide
API
Demo
A Development UI for ViewComponent
Lookbook is a tool to help develop, test and document
ViewComponents in isolation from your main application.
View the demo
or
Get Started »
LOOKBOOK DEMO
A dangerous action
A primary action
A less important action
This is the title


README.md
ViewComponent::Storybook
The ViewComponent::Storybook gem provides Ruby api for writing stories describing View Components and allowing them to be previewed and tested in Storybook via its Server support.
Features
A Ruby DSL for writing Stories describing View Components
A Rails controller backend for Storybook Server compatible with Storybook Controls Addon parameters
Coming Soon: Rake tasks to watch View Components and Stories and trigger Storybook hot reloading

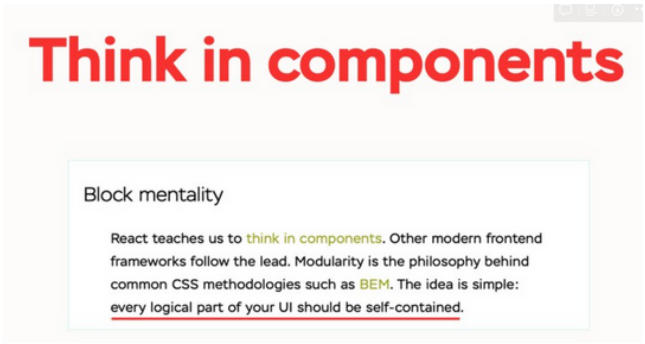

New
Storybook for React Native (6.5) ›
Automate with Chromatic
Storybook Day 2023
Storybook
Why
Showcase
Docs
Integrations
Community
Search docs ⌘K
Build UIs without the grunt work
Storybook is a frontend workshop for building UI components and pages in isolation. Thousands of teams use it for UI development, testing, and documentation. It's open source and free.
Get started
Watch video
v7
Latest version
18.29m
Installs per month
1944+
Contributors
localhost:8080
Storybook
Find by name
Introduction
Setup and configure
Changelog
LIBRARY
Charts
PieChart
SparkLine
RangeSlider
Overview
Default
No selection
Usage frequency


Projects that use Storybook
Storybook powers thousands of component libraries, design systems, and companies.
Filter by name
Popular
Chakra UI
Chakra 55
Fluent UI React
Microsoft 76
drei
pmndrs 69
shopify
Polaris React
Shopify 90
Kaizen
Culture Amp 58
shopify
Polaris Viz
Shopify 19
React-Dates
AirBnb 12
grommet
Grommet
Grommet 388
Fundamental Library
fundamental-styles
SAP 311
Grafana UI
Grafana 87
Fluent UI Web Components
Microsoft 38
Paste
Twilio 120
Carbon Components React
Carbon Design System 67
GitHub
Primer React
GitHub 80
JET BRAINS
Ring UI
JetBrains 122

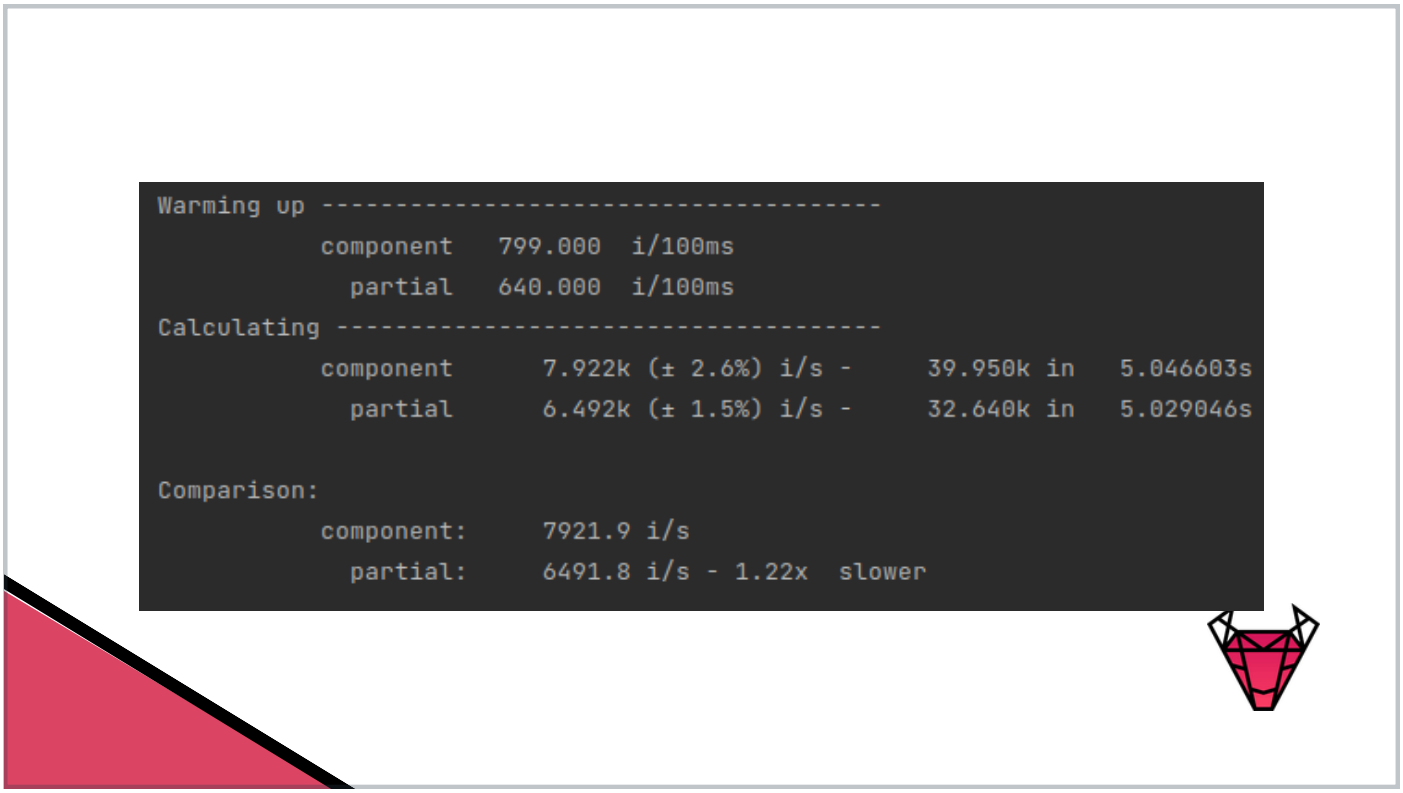
https://primer.style/view-components/lookbook/inspect/primer/alpha/form_control/playground



Dobre praktyki

1. Nie robić query w komponencie
2. Przekazywać tylko potrzebne dane(general-purpose, app-specific)
3. Zasada single responsibility
4. 

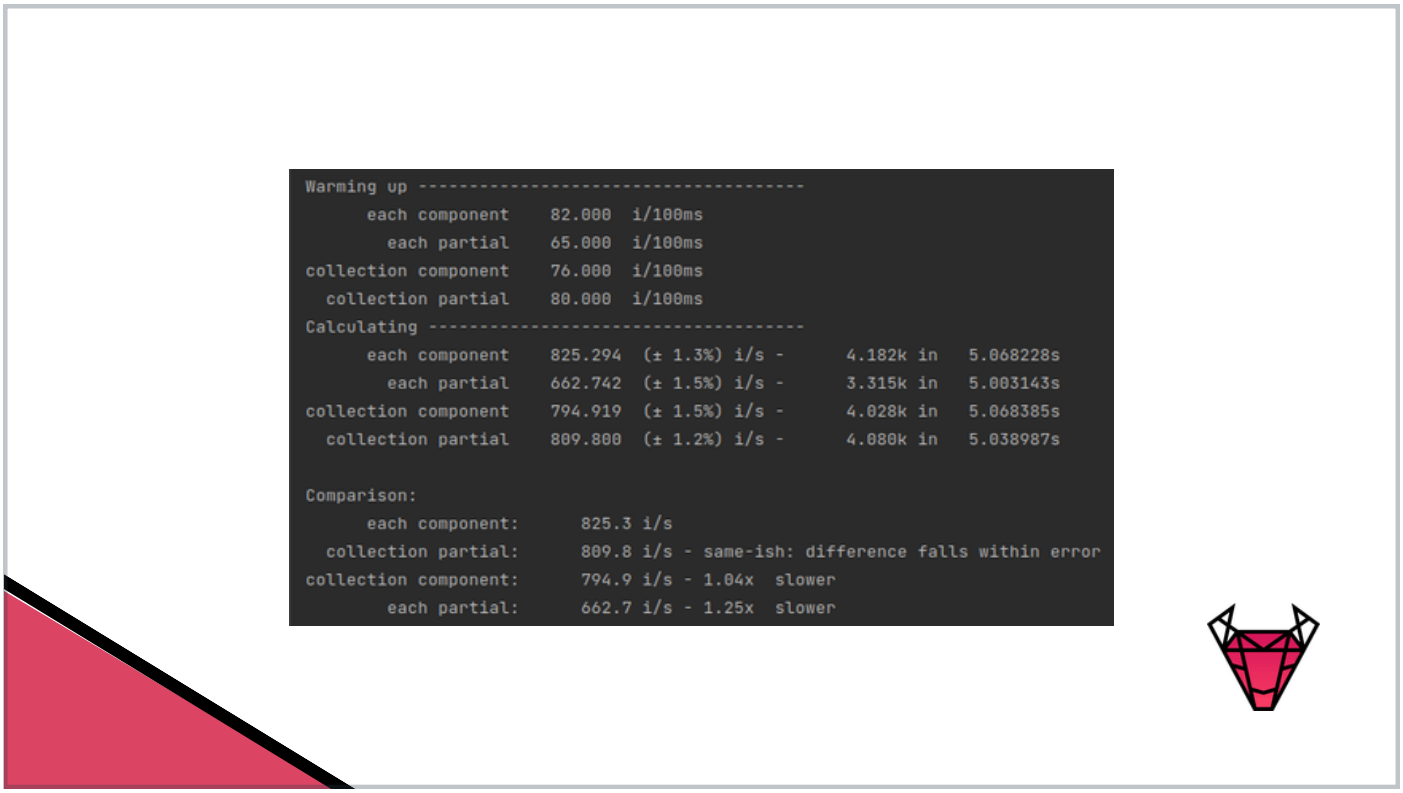
Monada to abstrakcyjny pojemnik na wartości, który udostępnia specjalne metody do przetwarzania tej wartości.



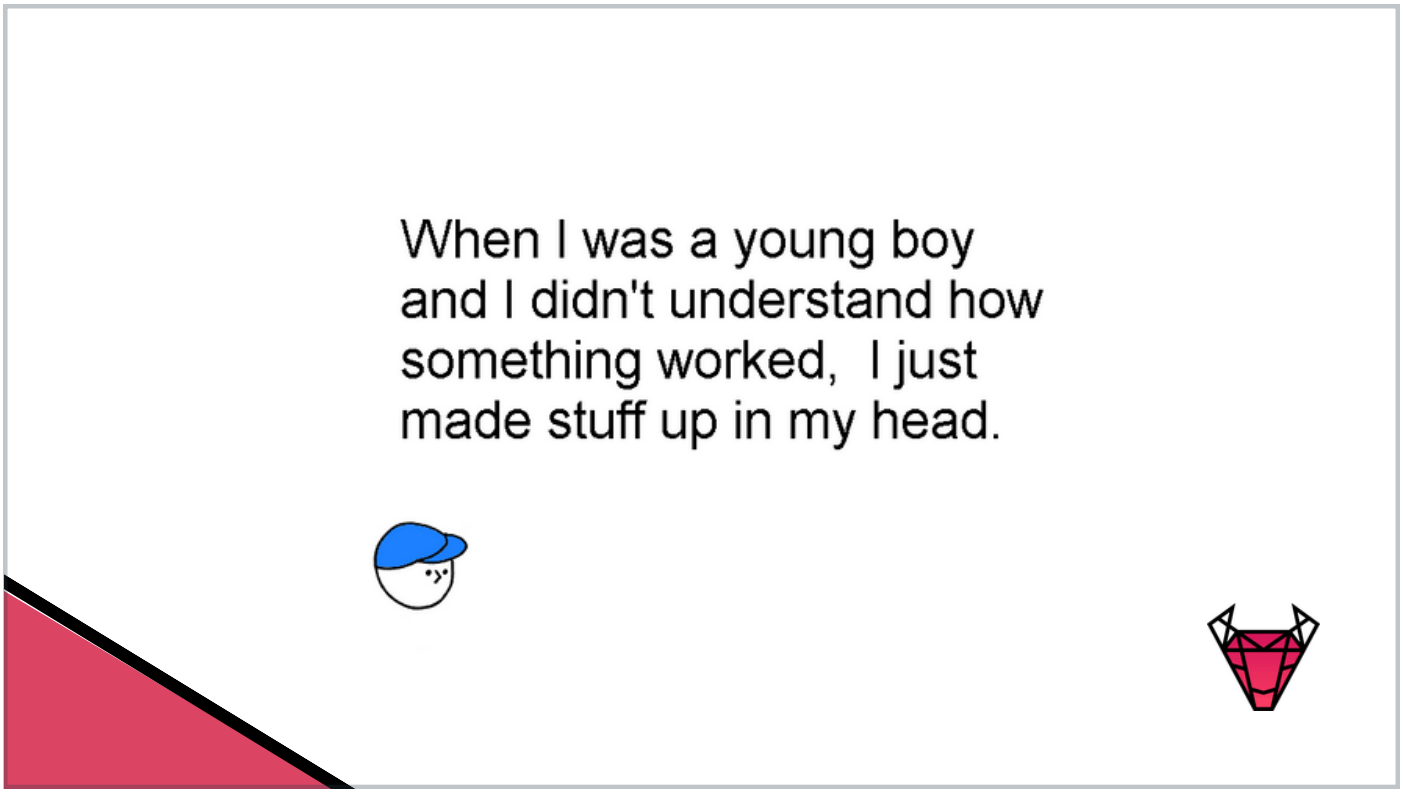
Monada to abstrakcyjny pojemnik na wartości, który udostępnia specjalne metody do przetwarzania tej wartości.

```ruby
Benchmark.ips do |x|
  x.report('each component') do
    employees.each do |employee|
      controller_view.render(Employees::CardIdComponent.new(employee:))
    end
  end
  x.report('each partial') do
    employees.each do |employee|
      controller_view.render('partial', employee:)
    end
  end
  x.report('collection component') do
    controller_view.render(Employees::CardIdComponent.with_collection(employees))
  end
  x.report('collection partial') do
    controller_view.render(partial: 'partial', collection: employees, as: :employee)
  end
  x.compare!
end
```

Monada to abstrakcyjny pojemnik na wartości, który udostępnia specjalne metody do przetwarzania tej wartości.



Monada to abstrakcyjny pojemnik na wartości, który udostępnia specjalne metody do przetwarzania tej wartości.



Monada to abstrakcyjny pojemnik na wartości, który udostępnia specjalne metody do przetwarzania tej wartości.

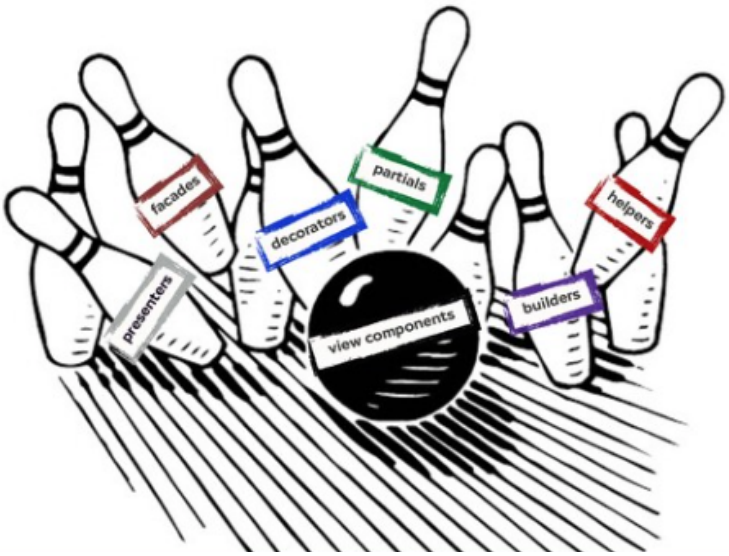
1. https://viewcomponent.org/
2. https://www.youtube.com/watch?v=YVYRus_2KZM
3. https://evilmartians.com/chronicles/viewcomponent-in-the-wild-building-modern-rails-frontends
4. https://evilmartians.com/chronicles/viewcomponent-in-the-wild-supercharging-your-components
5. https://noti.st/palkan/eVl0xO/frontendless-rails-frontend
6. https://evilmartians.com/opensource/view-component-contrib
7. https://primer.style/view-components/lookbook/inspect/primer/forms/immediate_validation_form/
8. https://dev.to/nejremeslnici/from-partials-to-viewcomponents-writing-reusable-front-end-code-in-rails-1c9o
9. https://design.gitlab.com/patterns/forms
10. https://docs.gitlab.com/ee/development/fe_guide/view_component.html
11. https://www.monkeyuser.com/2017/qa-engineer-buying-bed/?sc=true&dir=random
12. https://www.youtube.com/watch?v=slxvxp7E0xg
13. https://abstrusegoose.com/120
14. https://dev.to/abeidahmed/advanced-viewcomponent-patterns-in-rails-2b4m
15. https://mixandgo.com/learn/ruby-on-rails/view-component
16. https://www.codewithjason.com/different-kinds-rails-tests-use/

Monada to abstrakcyjny pojemnik na wartości, który udostępnia specjalne metody do przetwarzania tej wartości.

Posumowanie

Nie ważne, czy tworzymy aplikację SPA, czy monolith. Warto stosować komponenty.
Oba podejścia potrzebują ustrukturyzowanych widoków.
Przy najnowszym stacku technologicznym hotwire
view componenty stają się jeszcze przydatniejsze i przyjemniejsze do korzystania.

View Components



https://noti.st/palkan/eVl0xO/frontendless-rails-frontend

Nie ważne, czy tworzymy aplikację SPA, czy monolith. Warto stosować komponenty.
Oba podejścia potrzebują ustrukturyzowanych widoków.
Przy najnowszym stacku technologicznym hotwire
view componenty stają się jeszcze przydatniejsze i przyjemniejsze do korzystania.

ViewComponent - Czy to przyszłość tworzenia widoków w Ruby on Rails?

Thank you!

Pytania?