OTYŁOŚĆ

Alicja Cyganiewicz

alicja.cyganiewicz@infakt.pl

KRUG #6 / 2017

powered by Railsware.pl

17.10.2017 r.

Prawie **trzy czwarte** amerykańskich mężczyzn i ponad **60 procent** kobiet cierpi na otyłość lub nadwagę

Ryzyko chorób **serca**,
problemy z **oddychaniem**,
zwyrodnienia **stawów**...

Otyłość co roku **zabija** 300-400 tys. Amerykanów

OTYŁOŚĆ ?

OTYŁOŚĆ

245 kcal

```
class BooksController < < ActionController::Base
  def edit
    js_data = {}
    content = nil
    is_suspected = false
    action = symbolized_document_action(params[:a])
    if available_accountings?
      objects = ::Loader.new(current_user, params[:o]).objects
      is_suspected = objects.any? do |object|
        ::Books::Events::Verifier.new(object, action, date: params[:date]).blocked?
      end
      case
      when is_suspected && current_limits.cad_add_books?
        fid = "#{Time.now.to_i}#{current_user.id}".to_sha1[-6..-1]
        text = t(action, scope: 'flash.accountings.verify', default: :default)
        js_data['CorrectionBox'] = fid
        content = render_to_string(partial: ,/shared/box/partials/suspected',
                                   locals: { fid: fid, text: text }).strip
      when is_suspected
        flash[:error] = t('flash.accountings.verify.blocked')
        js_data['reload'] = true
      end
    end
    render json: { status: 'ok', suspected: is_suspected,  content: content, jsData: js_data }
  end

  def update
    scope = current_user.books.not_kind(:draft).not_kind(:sketch)
            .order("books.data_wystawienia ASC, books.id ASC").includes(:kind)
    if params[:search].blank? || params[:search].all? { |k, v| v.blank? }
      @search = scope.search({})
      @books = []
      @currency = 'PLN'
    else
      search = params[:search]
      @currency = search[:waluta_eq_any]
      search[:waluta_eq_any] = '' if search[:waluta_eq_any] == 'ALL'

      if search[:status_eq_any].present? && search[:status_eq_any].include?('booked')
        search[:status_eq_any] += %w[wydrukowana wysÅ‚ana]
        search[:status_eq_any].uniq!
      end
      @search = scope.search(params[:search])

      @books = @search.result(distinct: true)
    end
  end
end
```

Model

View

Controller

Logika Biznesowa

business logic

Model

View

Controller

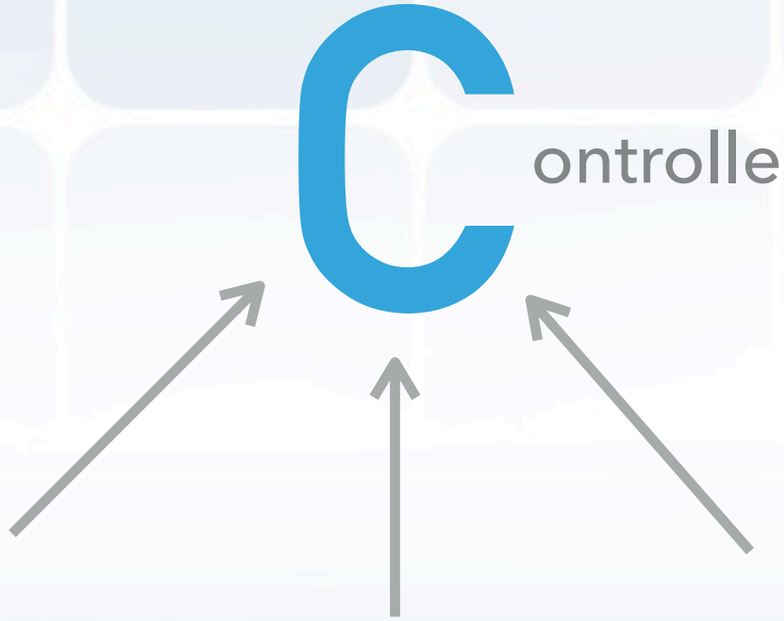
Logika Biznesowa

business logic

Model

View

Controller



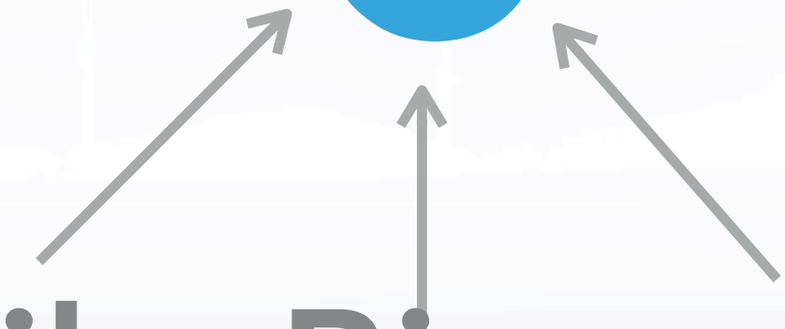
Logika Biznesowa

business logic

Model

View

Controller



Logika Biznesowa

business logic

Model

View

Controller

Logika Biznesowa

business logic

Model

View

Logika Biznesowa

Controller

business logic

Alicja Cyganiewicz

Ruby Dev at inFakt

alicja.cyganiewicz@infakt.pl

Alicja Cyganiewicz

Ruby Dev at **inFakt**

alicja.cyganiewicz@infakt.pl

księgowość

Alicja Cyganiewicz

Ruby Dev at **inFakt**

alicja.cyganiewicz@infakt.pl

księgowość

logika **biznesowa**

Choroby serca

Problemy z oddychaniem

Zwyrodnienia stawów

Tests...

DRY...?

```
class BooksController < < ActionController::Base
  def edit
    js_data = {}
    content = nil
    is_suspected = false
    action = symbolized_document_action(params[:a])
    if available_accountings?
      objects = ::Loader.new(current_user, params[:o]).objects
      is_suspected = objects.any? do |object|
        ::Books::Events::Verifier.new(object, action, date: params[:date]).blocked?
      end
      case
      when is_suspected && current_limits.cad_add_books?
        fid = "#{Time.now.to_i}#{current_user.id}".to_sha1[-6..-1]
        text = t(action, scope: 'flash.accountings.verify', default: :default)
        js_data['CorrectionBox'] = fid
        content = render_to_string(partial: ,/shared/box/partials/suspected',
                                   locals: { fid: fid, text: text }).strip
      when is_suspected
        flash[:error] = t('flash.accountings.verify.blocked')
        js_data['reload'] = true
      end
    end
    render json: { status: 'ok', suspected: is_suspected,  content: content, jsData: js_data }
  end

  def update
    scope = current_user.books.not_kind(:draft).not_kind(:sketch)
          .order("books.data_wystawienia ASC, books.id ASC").includes(:kind)
    if params[:search].blank? || params[:search].all? { |k, v| v.blank? }
      @search = scope.search({})
      @books = []
      @currency = 'PLN'
    else
      search = params[:search]
      @currency = search[:waluta_eq_any]
      search[:waluta_eq_any] = '' if search[:waluta_eq_any] == 'ALL'

      if search[:status_eq_any].present? && search[:status_eq_any].include?('booked')
        search[:status_eq_any] += %w[wydrukowana wysÅ‚ana]
        search[:status_eq_any].uniq!
      end
      @search = scope.search(params[:search])

      @books = @search.result(distinct: true)
    end
  end
end
```

Maintainable

Ponowne wykonanie logiki

np. API

DRY...?

Ponowne wykonanie logiki

np. API

DRY...?

D ont

R epeat

Y ourself

Tests

controller.rb

controller_spec.rb

```ruby
class BooksController < < ActionController::Base
  def edit
    js_data = {}
    content = nil
    is_suspected = false
    action = symbolized_document_action(params[:a])
    if available_accountings?
      objects = ::Loader.new(current_user, params[:o]).objects
      is_suspected = objects.any? do |object|
        ::Books::Events::Verifier.new(object, action, date: params[:date]).blocked?
      end
      case
      when is_suspected && current_limits.cad_add_books?
        fid = "#{Time.now.to_i}#{current_user.id}".to_sha1[-6..-1]
        text = t(action, scope: 'flash.accountings.verify', default: :default)
        js_data['CorrectionBox'] = fid
        content = render_to_string(partial: ,/shared/box/partials/suspected',
                                   locals: { fid: fid, text: text }).strip
      when is_suspected
        flash[:error] = t('flash.accountings.verify.blocked')
        js_data['reload'] = true
      end
    end
    render json: { status: 'ok', suspected: is_suspected,  content: content, jsData: js_data }
  end

  def update
    scope = current_user.books.not_kind(:draft).not_kind(:sketch)
            .order("books.data_wystawienia ASC, books.id ASC").includes(:kind)
    if params[:search].blank? || params[:search].all? { |k, v| v.blank? }
      @search = scope.search({})
      @books = []
      @currency = 'PLN'
    else
      search = params[:search]
      @currency = search[:waluta_eq_any]
      search[:waluta_eq_any] = '' if search[:waluta_eq_any] == 'ALL'

      if search[:status_eq_any].present? && search[:status_eq_any].include?('booked')
        search[:status_eq_any] += %w[wydrukowana wysÅ‚ana]
        search[:status_eq_any].uniq!
      end
      @search = scope.search(params[:search])

      @books = @search.result(distinct: true)
    end
  end
end
```


```ruby
class BooksController < < ActionController::Base
  def edit
    js_data = {}
    content = nil
    is_suspected = false
    action = symbolized_document_action(params[:a])
    if available_accountings?
      objects = ::Loader.new(current_user, params[:o]).objects
      is_suspected = objects.any? do |object|
        ::Books::Events::Verifier.new(object, action, date: params[:date]).blocked?
      end
      case
      when is_suspected && current_limits.cad_add_books?
        fid = "#{Time.now.to_i}#{current_user.id}".to_sha1[-6..-1]
        text = t(action, scope: 'flash.accountings.verify', default: :default)
        js_data['CorrectionBox'] = fid
        content = render_to_string(partial: ,/shared/box/partials/suspected',
                                   locals: { fid: fid, text: text }).strip
      when is_suspected
        flash[:error] = t('flash.accountings.verify.blocked')
        js_data['reload'] = true
      end
    end
    render json: { status: 'ok', suspected: is_suspected,  content: content, jsData: js_data }
  end

  def update
    scope = current_user.books.not_kind(:draft).not_kind(:sketch)
            .order("books.data_wystawienia ASC, books.id ASC").includes(:kind)
    if params[:search].blank? || params[:search].all? { |k, v| v.blank? }
      @search = scope.search({})
      @books = []
      @currency = 'PLN'
    else
      search = params[:search]
      @currency = search[:waluta_eq_any]
      search[:waluta_eq_any] = '' if search[:waluta_eq_any] == 'ALL'

      if search[:status_eq_any].present? && search[:status_eq_any].include?('booked')
        search[:status_eq_any] += %w[wydrukowana wysÅ‚ana]
        search[:status_eq_any].uniq!
      end
      @search = scope.search(params[:search])

      @books = @search.result(distinct: true)
    end
  end
end
```

controller.rb

controller_spec.rb

Tests

```
class BooksController < < ActionController::Base
  def edit
    js_data = {}
    content = nil
    is_suspected = false
    action = symbolized_document_action(params[:a])
    if available_accountings?
      objects = ::Loader.new(current_user, params[:o]).objects
      is_suspected = objects.any? do |object|
        ::Books::Events::Verifier.new(object, action, date: params[:date]).blocked?
      end
      case
      when is_suspected && current_limits.cad_add_books?
        fid = "#{Time.now.to_i}#{current_user.id}".to_sha1[-6..-1]
        text = t(action, scope: 'flash.accountings.verify', default: :default)
        js_data['CorrectionBox'] = fid
        content = render_to_string(partial: ,/shared/box/partials/suspected',
                                   locals: { fid: fid, text: text }).strip
      when is_suspected
        flash[:error] = t('flash.accountings.verify.blocked')
        js_data['reload'] = true
      end
    end
    render json: { status: 'ok', suspected: is_suspected,  content: content, jsData: js_data }
  end

  def update
    scope = current_user.books.not_kind(:draft).not_kind(:sketch)
            .order("books.data_wystawienia ASC, books.id ASC").includes(:kind)
    if params[:search].blank? || params[:search].all? { |k, v| v.blank? }
      @search = scope.search({})
      @books = []
      @currency = 'PLN'
    else
      search = params[:search]
      @currency = search[:waluta_eq_any]
      search[:waluta_eq_any] = '' if search[:waluta_eq_any] == 'ALL'

      if search[:status_eq_any].present? && search[:status_eq_any].include?('booked')
        search[:status_eq_any] += %w[wydrukowana wysÅ‚ana]
        search[:status_eq_any].uniq!
      end
      @search = scope.search(params[:search])

      @books = @search.result(distinct: true)
    end
  end
end
```

stub_all

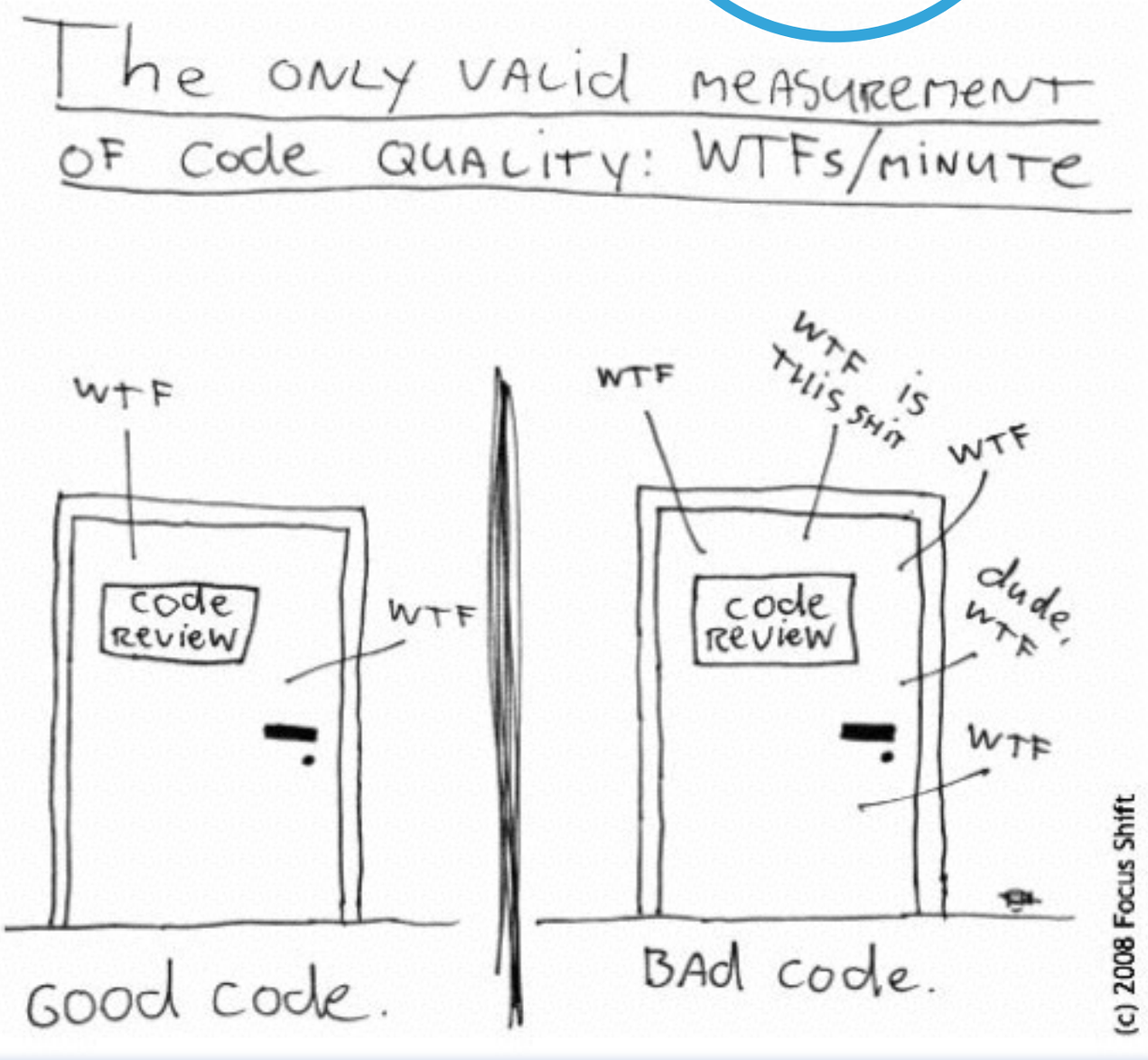
test_nothing

Kompletnie nieczytelny kod

Nierozszerzalny

low maintainability

Maintainable



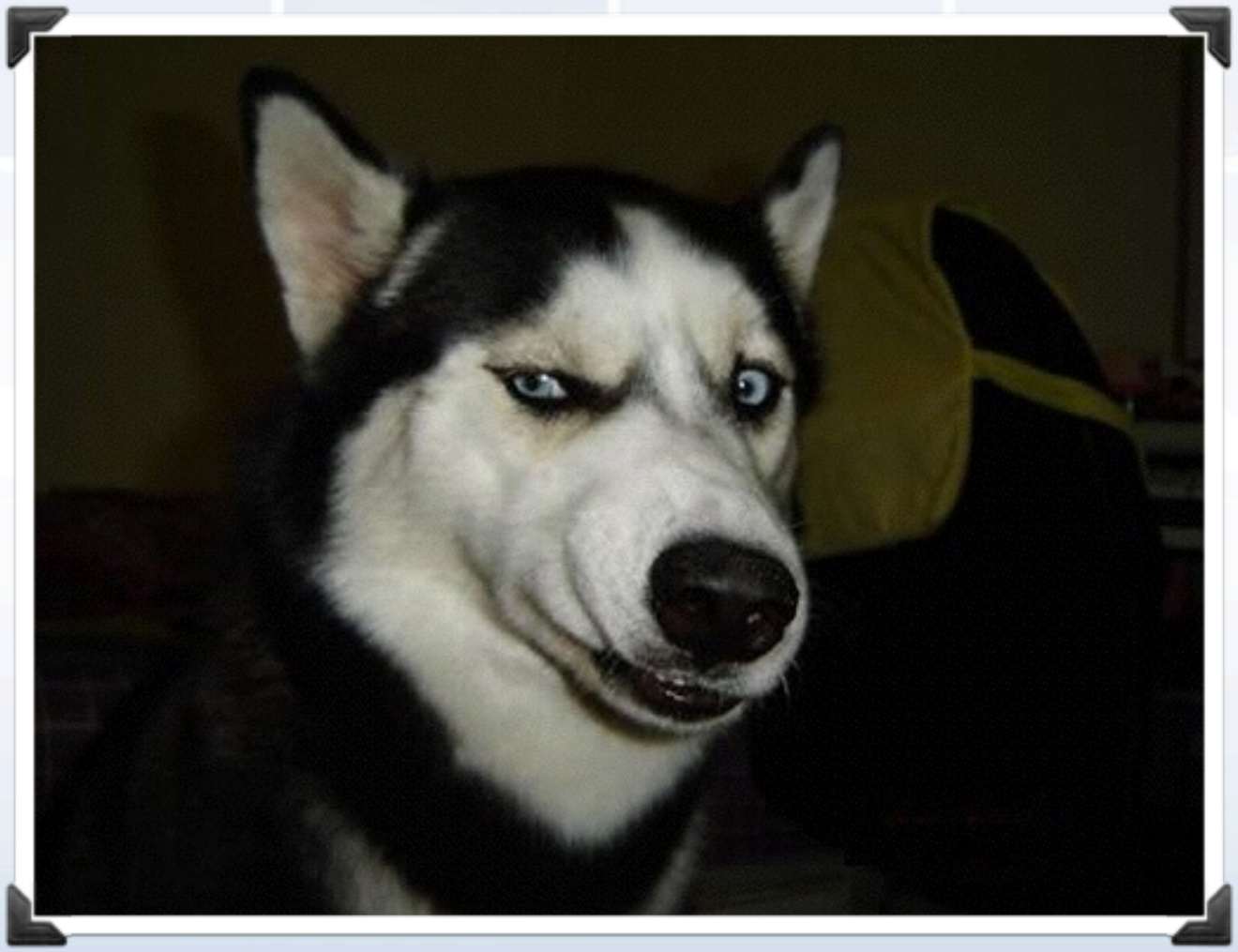
There was a **time**...

There was a **time**...

Skinny controllers, **FAT** models

Skinny controllers, FAT models

Business logic in models?



So were we are?

FAT controllers

```
class BooksController < < ActionController::Base
  def edit
    js_data = {}
    content = nil
    is_suspected = false
    action = symbolized_document_action(params[:a])
    if available_accountings?
      objects = ::Loader.new(current_user, params[:o]).objects
      is_suspected = objects.any? do |object|
        ::Books::Events::Verifier.new(object, action, date: params[:date]).blocked?
      end
      case
      when is_suspected && current_limits.cad_add_books?
        fid = "#{Time.now.to_i}#{current_user.id}".to_sha1[-6..-1]
        text = t(action, scope: 'flash.accountings.verify', default: :default)
        js_data['CorrectionBox'] = fid
        content = render_to_string(partial: ,/shared/box/partials/suspected',
                                   locals: { fid: fid, text: text }).strip
      when is_suspected
        flash[:error] = t('flash.accountings.verify.blocked')
        js_data['reload'] = true
      end
    end
    render json: { status: 'ok', suspected: is_suspected,  content: content, jsData: js_data }
  end

  def update
    scope = current_user.books.not_kind(:draft).not_kind(:sketch)
            .order("books.data_wystawienia ASC, books.id ASC").includes(:kind)
    if params[:search].blank? || params[:search].all? { |k, v| v.blank? }
      @search = scope.search({})
      @books = []
      @currency = 'PLN'
    else
      search = params[:search]
      @currency = search[:waluta_eq_any]
      search[:waluta_eq_any] = '' if search[:waluta_eq_any] == 'ALL'

      if search[:status_eq_any].present? && search[:status_eq_any].include?('booked')
        search[:status_eq_any] += %w[wydrukowana wysÅ‚ana]
        search[:status_eq_any].uniq!
      end
      @search = scope.search(params[:search])

      @books = @search.result(distinct: true)
    end
  end
end
```

So were we are?

FAT controllers

```
class BooksController < < ActionController::Base
  def edit
    js_data = {}
    content = nil
    is_suspected = false
    action = symbolized_document_action(params[:a])
    if available_accountings?
      objects = ::Loader.new(current_user, params[:o]).objects
      is_suspected = objects.any? do |object|
        ::Books::Events::Verifier.new(object, action, date: params[:date]).blocked?
      end
      case
      when is_suspected && current_limits.cad_add_books?
        fid = "#{Time.now.to_i}#{current_user.id}".to_sha1[-6..-1]
        text = t(action, scope: 'flash.accountings.verify', default: :default)
        js_data['CorrectionBox'] = fid
        content = render_to_string(partial: ,/shared/box/partials/suspected',
                                   locals: { fid: fid, text: text }).strip
      when is_suspected
        flash[:error] = t('flash.accountings.verify.blocked')
        js_data['reload'] = true
      end
    end
    render json: { status: 'ok', suspected: is_suspected,  content: content, jsData: js_data }
  end

  def update
    scope = current_user.books.not_kind(:draft).not_kind(:sketch)
            .order("books.data_wystawienia ASC, books.id ASC").includes(:kind)
    if params[:search].blank? || params[:search].all? { |k, v| v.blank? }
      @search = scope.search({})
      @books = []
      @currency = 'PLN'
    else
      search = params[:search]
      @currency = search[:waluta_eq_any]
      search[:waluta_eq_any] = '' if search[:waluta_eq_any] == 'ALL'

      if search[:status_eq_any].present? && search[:status_eq_any].include?('booked')
        search[:status_eq_any] += %w[wydrukowana wysÅ‚ana]
        search[:status_eq_any].uniq!
      end
      @search = scope.search(params[:search])

      @books = @search.result(distinct: true)
    end
  end
end
```

Model

View

Controller

So were we are?

FAT controllers

```
class BooksController < < ActionController::Base
  def edit
    js_data = {}
    content = nil
    is_suspected = false
    action = symbolized_document_action(params[:a])
    if available_accountings?
      objects = ::Loader.new(current_user, params[:o]).objects
      is_suspected = objects.any? do |object|
        ::Books::Events::Verifier.new(object, action, date: params[:date]).blocked?
      end
      case
      when is_suspected && current_limits.cad_add_books?
        fid = "#{Time.now.to_i}#{current_user.id}".to_sha1[-6..-1]
        text = t(action, scope: 'flash.accountings.verify', default: :default)
        js_data['CorrectionBox'] = fid
        content = render_to_string(partial: ,/shared/box/partials/suspected',
                                   locals: { fid: fid, text: text }).strip
      when is_suspected
        flash[:error] = t('flash.accountings.verify.blocked')
        js_data['reload'] = true
      end
    end
    render json: { status: 'ok', suspected: is_suspected,  content: content, jsData: js_data }
  end

  def update
    scope = current_user.books.not_kind(:draft).not_kind(:sketch)
            .order("books.data_wystawienia ASC, books.id ASC").includes(:kind)
    if params[:search].blank? || params[:search].all? { |k, v| v.blank? }
      @search = scope.search({})
      @books = []
      @currency = 'PLN'
    else
      search = params[:search]
      @currency = search[:waluta_eq_any]
      search[:waluta_eq_any] = '' if search[:waluta_eq_any] == 'ALL'

      if search[:status_eq_any].present? && search[:status_eq_any].include?('booked')
        search[:status_eq_any] += %w[wydrukowana wysÅ‚ana]
        search[:status_eq_any].uniq!
      end
      @search = scope.search(params[:search])

      @books = @search.result(distinct: true)
    end
  end
end
```

Model

View

Controller

Homeless business logic

SERVICES

OBJECTS

SLIMMING THERAPY

FAT controllers

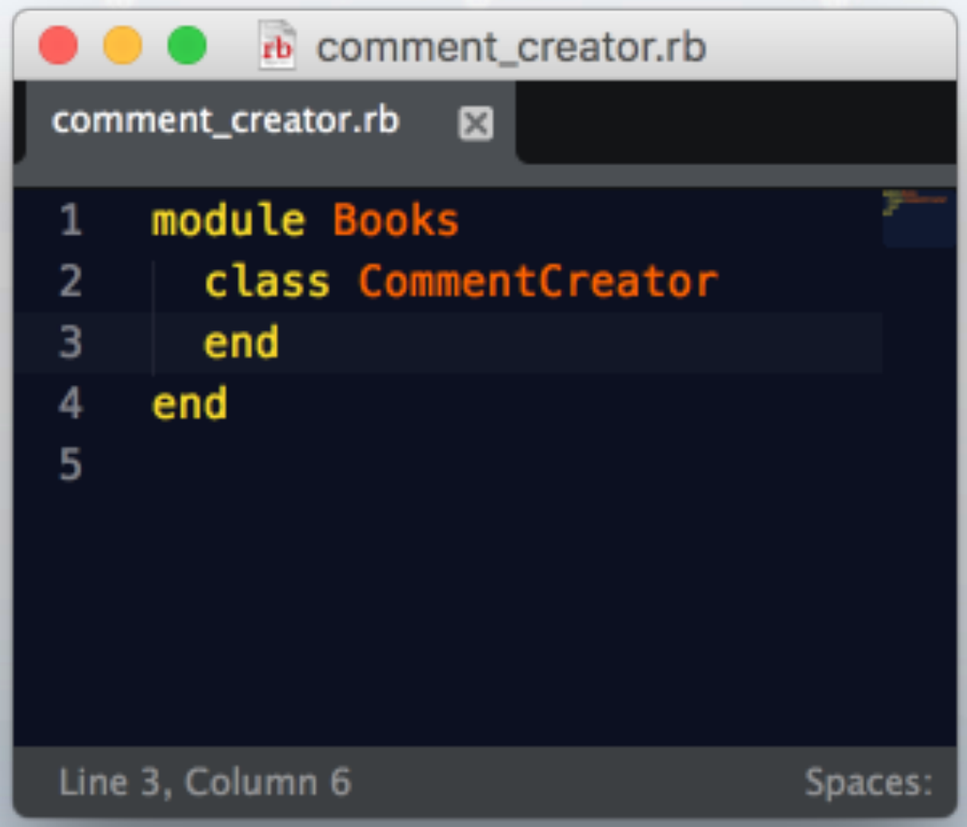
```
class BooksController < < ActionController::Base
  def edit
    js_data = {}
    content = nil
    is_suspected = false
    action = symbolized_document_action(params[:a])
    if available_accountings?
      objects = ::Loader.new(current_user, params[:o]).objects
      is_suspected = objects.any? do |object|
        ::Books::Events::Verifier.new(object, action, date: params[:date]).blocked?
      end
      case
      when is_suspected && current_limits.cad_add_books?
        fid = "#{Time.now.to_i}#{current_user.id}".to_sha1[-6..-1]
        text = t(action, scope: 'flash.accountings.verify', default: :default)
        js_data['CorrectionBox'] = fid
        content = render_to_string(partial: ,/shared/box/partials/suspected',
                                   locals: { fid: fid, text: text }).strip
      when is_suspected
        flash[:error] = t('flash.accountings.verify.blocked')
        js_data['reload'] = true
      end
    end
    render json: { status: 'ok', suspected: is_suspected,  content: content, jsData: js_data }
  end

  def update
    scope = current_user.books.not_kind(:draft).not_kind(:sketch)
            .order("books.data_wystawienia ASC, books.id ASC").includes(:kind)
    if params[:search].blank? || params[:search].all? { |k, v| v.blank? }
      @search = scope.search({})
      @books = []
      @currency = 'PLN'
    else
      search = params[:search]
      @currency = search[:waluta_eq_any]
      search[:waluta_eq_any] = '' if search[:waluta_eq_any] == 'ALL'

      if search[:status_eq_any].present? && search[:status_eq_any].include?('booked')
        search[:status_eq_any] += %w[wydrukowana wysÅ‚ana]
        search[:status_eq_any].uniq!
      end
      @search = scope.search(params[:search])

      @books = @search.result(distinct: true)
    end
  end
end
```

app

-- services

---- books

------ comment_creator.rb



app

–– services

––––– books

––––––– comment_creator.rb

comment_creator.rb

```ruby
module Books
  class CommentCreator
    def initialize(user: nil, data: nil)
      @user = user
      @data = data
    end

    def perform
      #perform whole logic from controller
    end
  end
end

```

Line 10, Column 8 Spaces: 2 Ruby

comments_controller.rb

comments_controller.rb

```ruby
module Books
  class CommentsController < ActionController::Base
    #...

    def create
      book = Book.find(params[:id])
      if book && valid_form?
        if current_user
          nick = current_user.login
        else
          nick = "gosc" + Time.now.to_s
        end
        if book.comments.create(comment_data.merge(author: nick))
          book.followers.each do |followe|
            ::Books::CommentNotifier.seliver(follower)
          end
          Books::RateCalculatorWorker.perform_asyn(book.id)
          redirect_to book_path(book)
        else
          #...
```

Line 29, Column 4 Spaces: 2 Ruby

comments_controller.rb

comments_controller.rb

```
1  module Books
2    class CommentsController < ActionController::Base
3      #...
4
5      def create
6        if ::Books::CommentCreator
```

Syntax error, unexpected end-of-input, Line 6, Column 33 Spaces: 2 Ruby

comments_controller.rb

comments_controller.rb

```ruby
module Books
  class CommentsController < ActionController::Base
    #...

    def create
      if ::Books::CommentCreator.new(user: current_user, data: params)
```

Syntax error, unexpected end-of-input, Line 6, Column 71 Spaces: 2 Ruby

comments_controller.rb

comments_controller.rb

```ruby
module Books
  class CommentsController < ActionController::Base
    #...

    def create
      if ::Books::CommentCreator.new(user: current_user, data: params)
        .perform

```

Syntax error, unexpected end-of-input, Line 6, Column 79 Spaces: 2 Ruby

comments_controller.rb

comments_controller.rb

```ruby
module Books
  class CommentsController < ActionController::Base
    #...

    def create
      if ::Books::CommentCreator.new(user: current_user, data: params)
        .perform
        redirect_to book_path(book)
      else
        #...
```

Line 9, Column 13 Spaces: 2 Ruby

comments_controller.rb

comments_controller.rb

```ruby
module Books
  class CommentsController < ActionController::Base
    #...

    def create
      book = Book.find(params[:id])
      if book && valid_form?
        if current_user
          nick = current_user.login
        else
          nick = "gosc" + Time.now.to_s
        end
        if book.comments.create(comment_data.merge(author: nick))
          book.followers.each do |followe|
            ::Books::CommentNotifier.seliver(follower)
          end
          Books::RateCalculatorWorker.perform_asyn(book.id)
          redirect_to book_path(book)
        else
          #...
```

11 LINES

Line 29, Column 4 Spaces: 2 Ruby

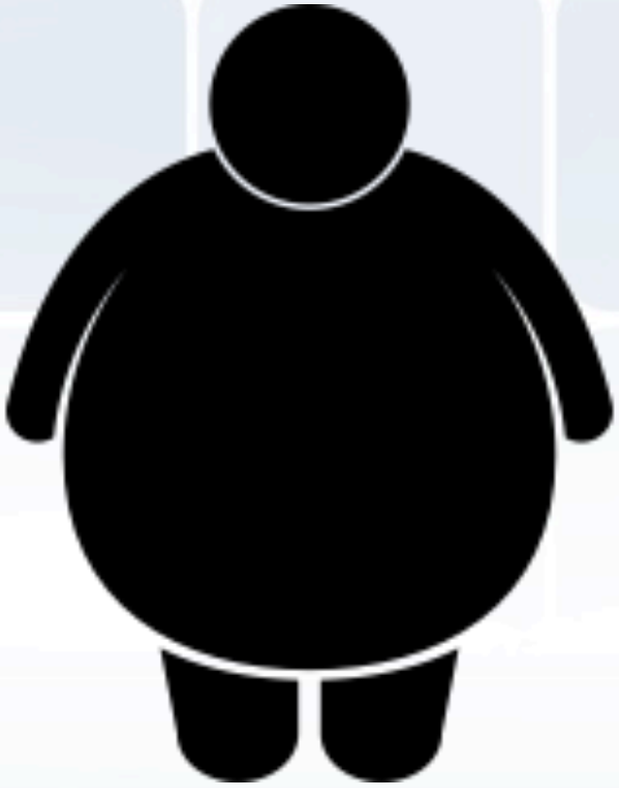
comments_controller.rb

comments_controller.rb

```ruby
module Books
  class CommentsController < ActionController::Base
    #...

    def create
      if ::Books::CommentCreator.new(user: current_user, data: params)
        .perform
        redirect_to book_path(book)
      else
        #...
```

+1 LINES

Line 9, Column 13 Spaces: 2 Ruby



Services Objects

slimming therapy

comments_controller.rb

```ruby
module Books
  class CommentsController < ActionController::Base
    #...

    def create
      if ::Books::CommentCreator.new(user: current_user, data: params)
        .perform
        redirect_to book_path(book)
      else
        #...
```

Line 9, Column 13 Spaces: 2 Ruby

DRY...?

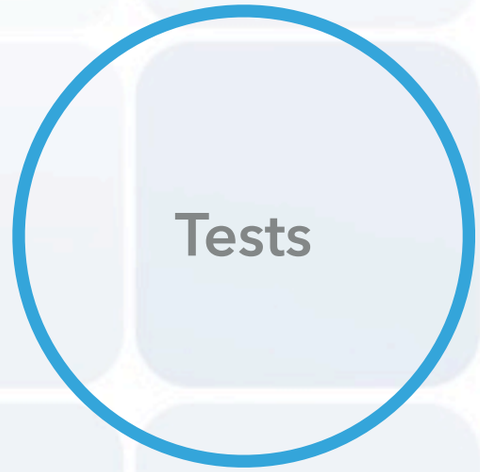
Service re-used!

comments_controller.rb

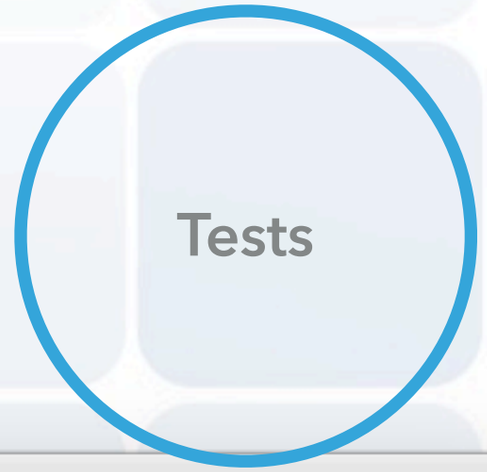
```ruby
module Api
  module Books
    class CommentsController < ::Api::BaseController
      #...

      def create
        if ::Books::CommentCreator.new(user: current_user, data:
          params).perform
          render json: { comment: comment }, status: :created
        else
```

Line 9, Column 13 Spaces: 2 Ruby

prosty test *controllera* testujący to, czego od niego oczekujemy



prosty test *controllera* testujący **to,**
czego od niego oczekujemy

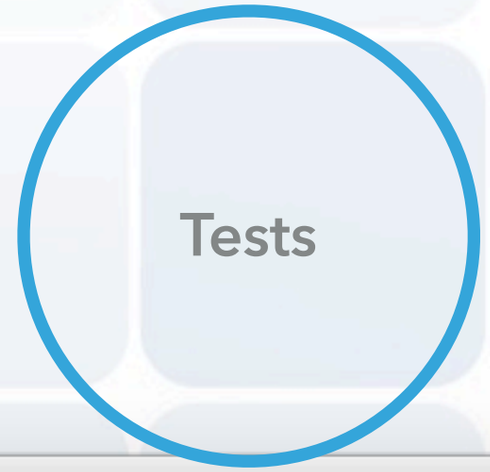


comments_controller_spec.rb

```ruby
require 'rails_helper'

RSpec.describe Books::CommentsController, type: :controller do
  #..
  describe 'POST #create' do
    context 'comment successfully created' do
      before {
        expect(::Book::CommentCreator).to receive_message_chain(:new, :perform).with(
          user: user, data: comment_params).with(no_args).and_return(true)
      }
      it 'redirects to book details' do
        post :create, params: comment_params
        expect(response).to have_http_status(:redirect)
      end
    end
```

Line 12, Column 56 Spaces: 2 Ruby

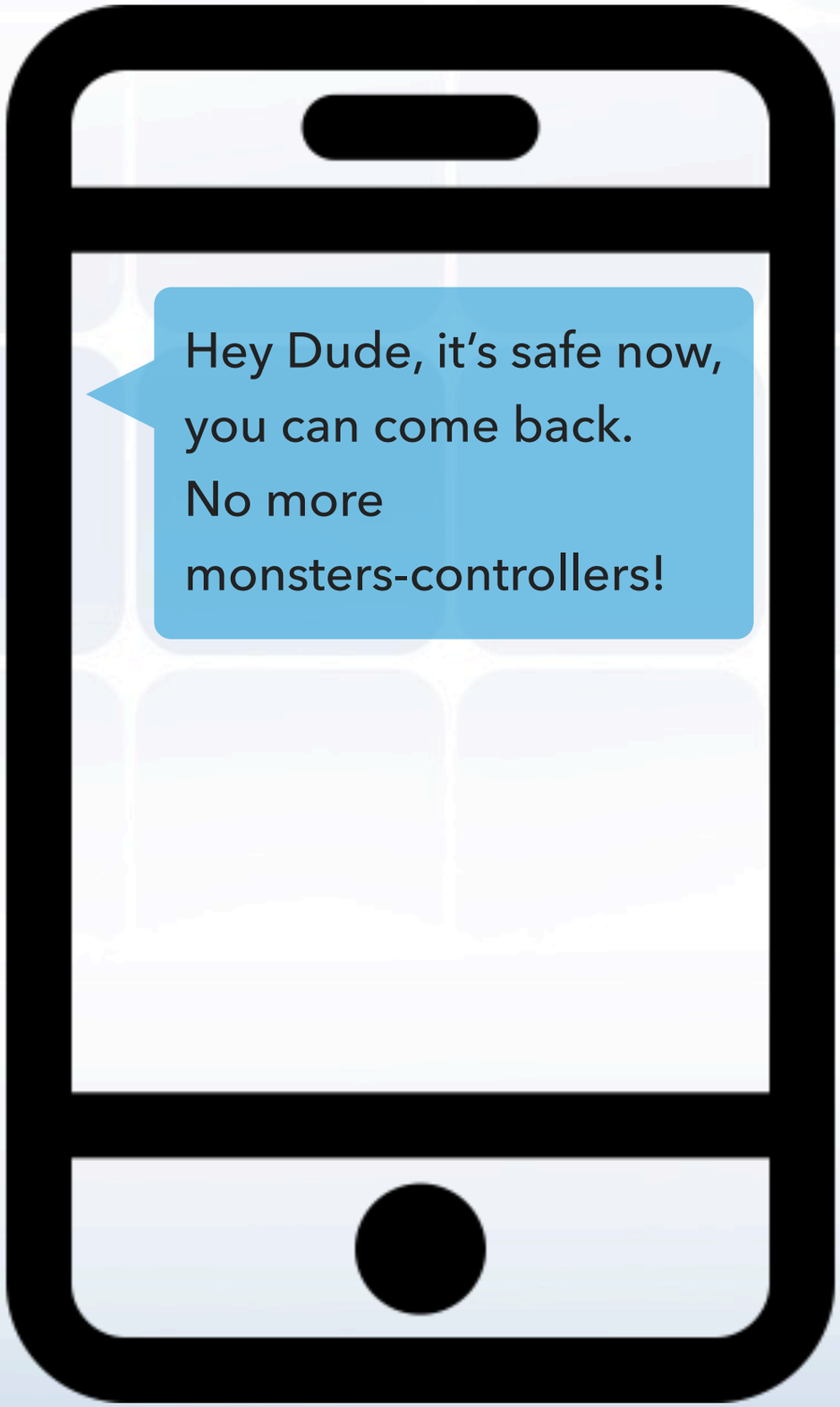
…a całą skomplikowaną logikę przetestujemy sobie *spokojnie* w teście serwisu

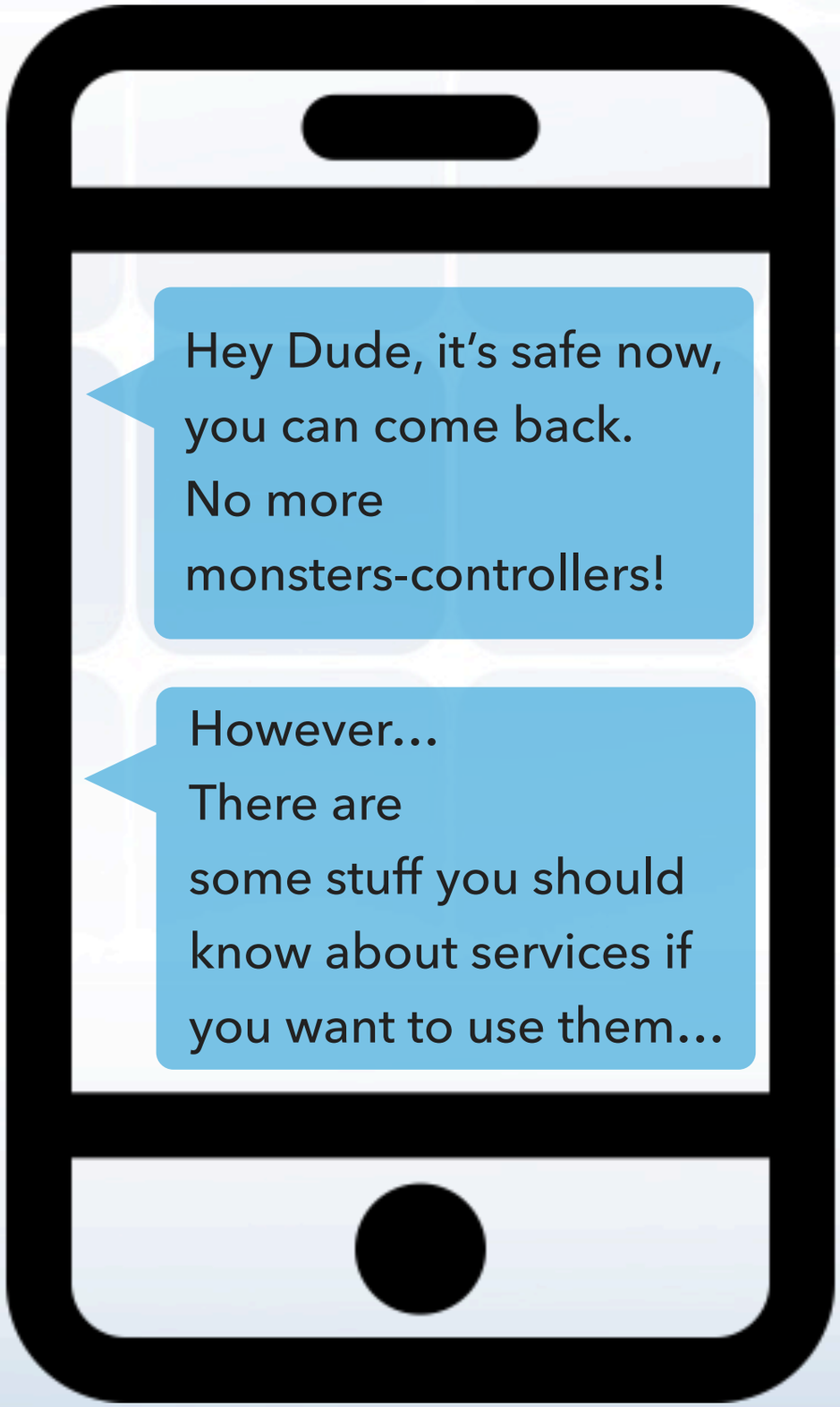


comment_creator_spec.rb

```ruby
require 'rails_helper'

RSpec.describe Books::CommentCreator do
  let(:user) { build(:user) }
  let(:data) { ... }
  #...
  subject { described_class.new(user: user, data: data) }

  describe '#perform' do
    context 'when form is valid' do
      #...
```

Line 10, Column 36 Spaces: 2 Ruby


Hey Dude, it’s safe now,
you can come back.
No more
monsters-controllers!


Hey Dude, it's safe now, you can come back. No more monsters-controllers!
However... There are some stuff you should know about services if you want to use them...

Plain

Old

Ruby

Object

SERVICES OBJECTS

RAILS

Single Responsibility Principle

Self - explanatory

naming

MoneyConverter,

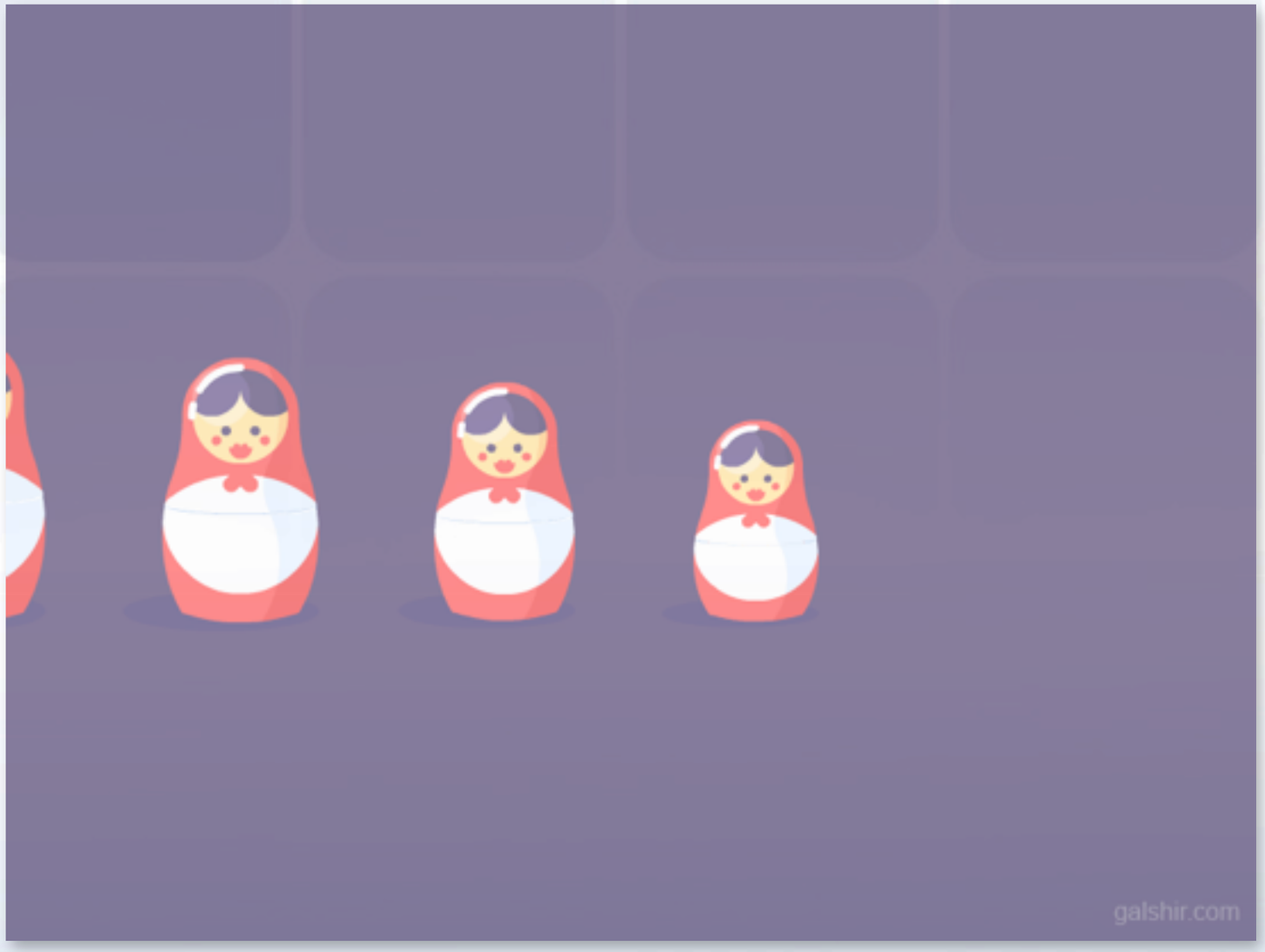
AgreementSigner,

EventFetcher...

One public method

SRP → .PERFORM

Services in a service



books_controller.rb

books_controller.rb

```ruby
class BooksController < ActionController::Base
  #...
  def update
    @book = Book.find(params[:id])
    if ::BookUpdater.new(@book, params).perform
      redirect_to book_path(@book)
    else
      render 'edit'
    end
  end
end
```

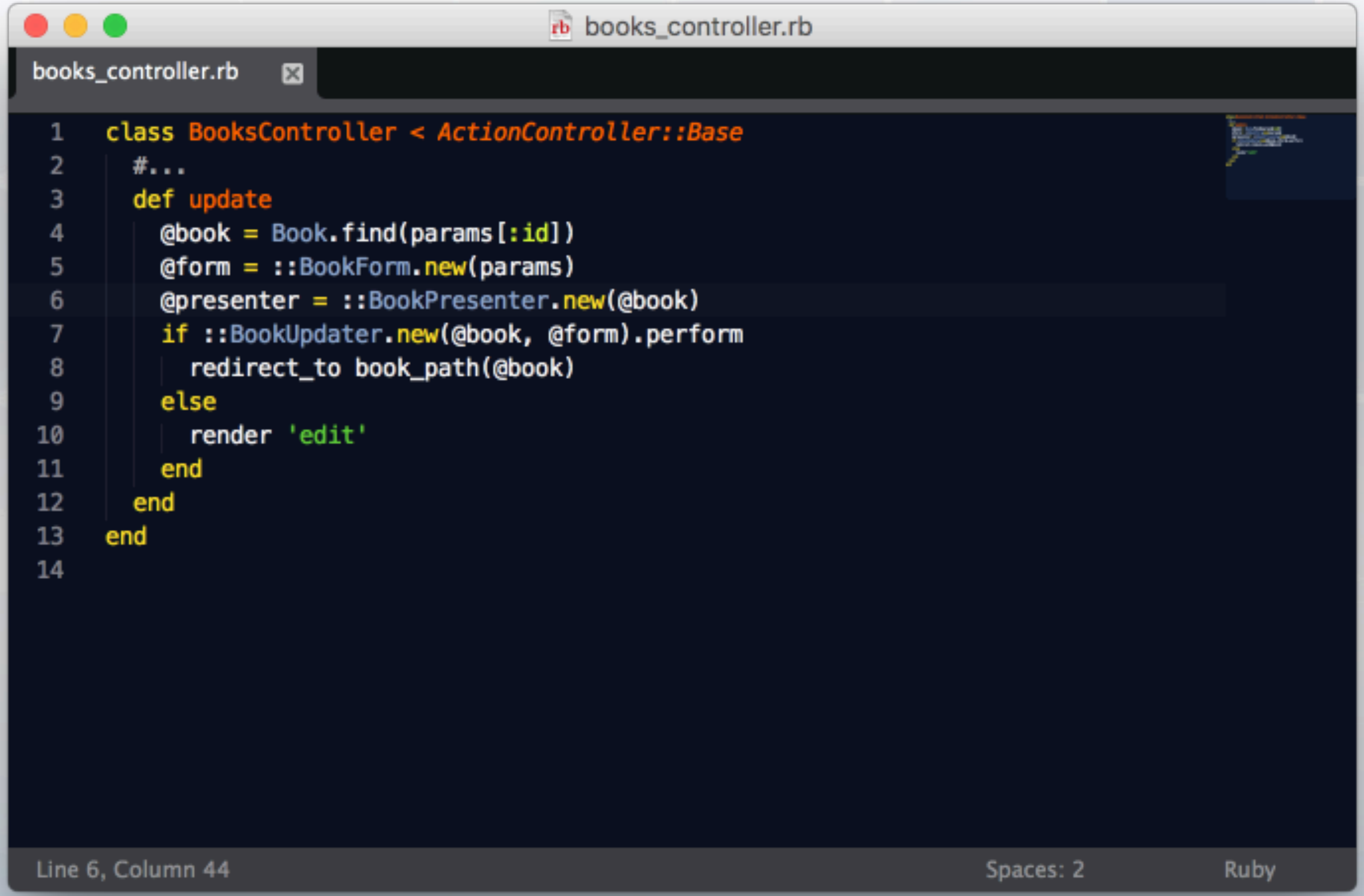
Line 1, Column 1 Spaces: 2 Ruby

books_controller.rb

books_controller.rb

```ruby
class BooksController < ActionController::Base
  #...
  def update
    @book = Book.find(params[:id])
    @form = ::BookForm.new(params)
    if ::BookUpdater.new(@book, @form).perform
      redirect_to book_path(@book)
    else
      render 'edit'
    end
  end
end

```

Line 5, Column 35 Spaces: 2 Ruby


books_controller.rb
books_controller.rb
class BooksController < ActionController::Base
#...
def update
@book = Book.find(params[:id])
@form = ::BookForm.new(params)
@presenter = ::BookPresenter.new(@book)
if ::BookUpdater.new(@book, @form).perform
redirect_to book_path(@book)
else
render 'edit'
end
end
end
Line 6, Column 44
Spaces: 2
Ruby

books_controller.rb

```ruby
class BooksController < ActionController::Base
  #...
  def update
    @book = Book.find(params[:id])
    @form = ::BookForm.new(params)
    @presenter = ::BookPresenter.new(@book)
    if ::BookUpdater.new(@book, @form).perform
      redirect_to book_path(@book)
    else
      if form.valid?
        render 'update_error'
      else
        render 'edit'
      end
    end
  end
end

```

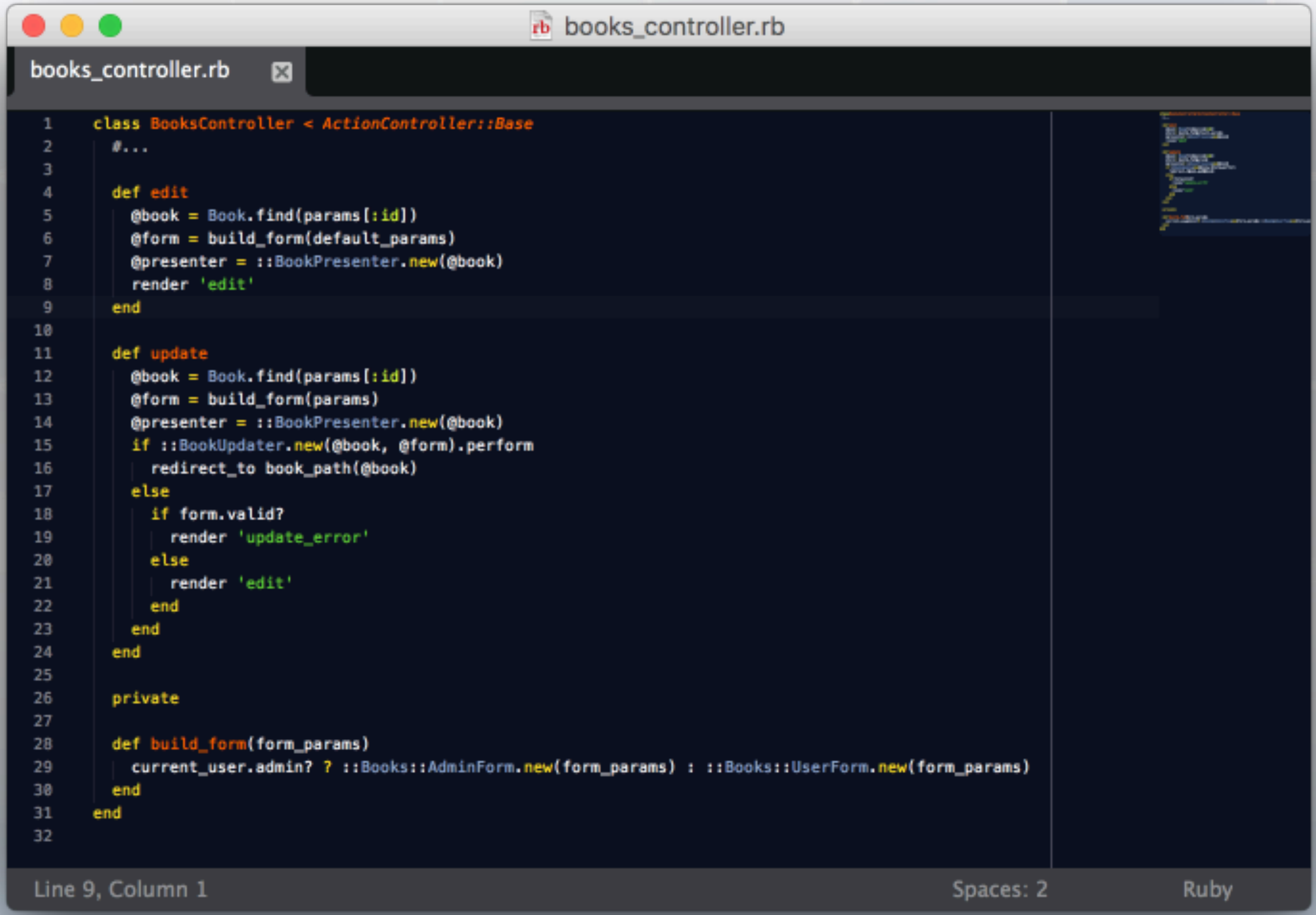
Line 14, Column 10 Spaces: 2 Ruby

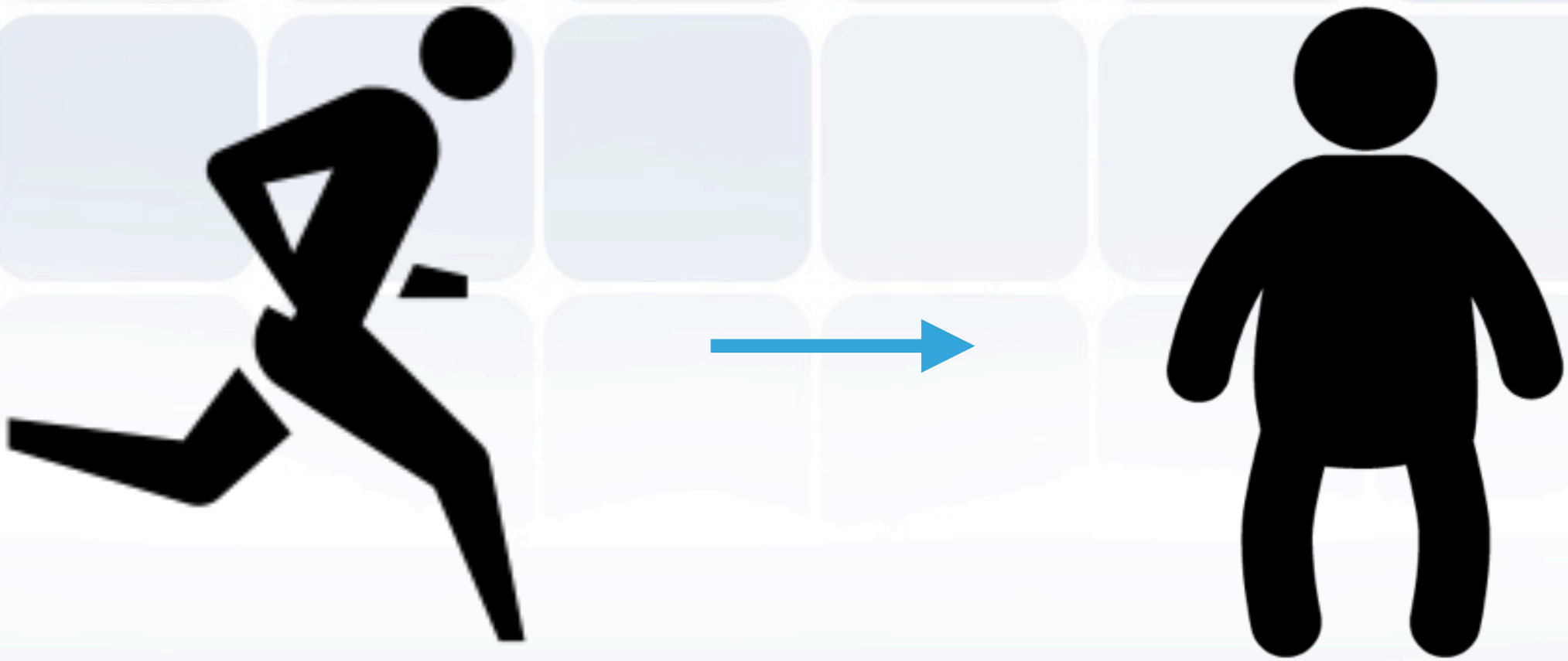
books_controller.rb

books_controller.rb

```ruby
class BooksController < ActionController::Base
  #...
  def update
    @book = Book.find(params[:id])
    @form = build_form(params)
    @presenter = ::BookPresenter.new(@book)
    if ::BookUpdater.new(@book, @form).perform
      redirect_to book_path(@book)
    else
      if form.valid?
        render 'update_error'
      else
        render 'edit'
      end
    end
  end

  private

  def build_form(form_params)
    current_user.admin? ? ::Books::AdminForm.new(form_params) : ::Books::
      UserForm.new(form_params)
  end
```

Line 10, Column 21 Spaces: 2 Ruby


books_controller.rb
books_controller.rb
class BooksController < ActionController::Base
#...
def edit
@book = Book.find(params[:id])
@form = build_form(default_params)
@presenter = ::BookPresenter.new(@book)
render 'edit'
end
def update
@book = Book.find(params[:id])
@form = build_form(params)
@presenter = ::BookPresenter.new(@book)
if ::BookUpdater.new(@book, @form).perform
redirect_to book_path(@book)
else
if form.valid?
render 'update_error'
else
render 'edit'
end
end
end
private
def build_form(form_params)
current_user.admin? ? ::Books::AdminForm.new(form_params) : ::Books::UserForm.new(form_params)
end
end
Line 9, Column 1
Spaces: 2
Ruby



PRESENTER

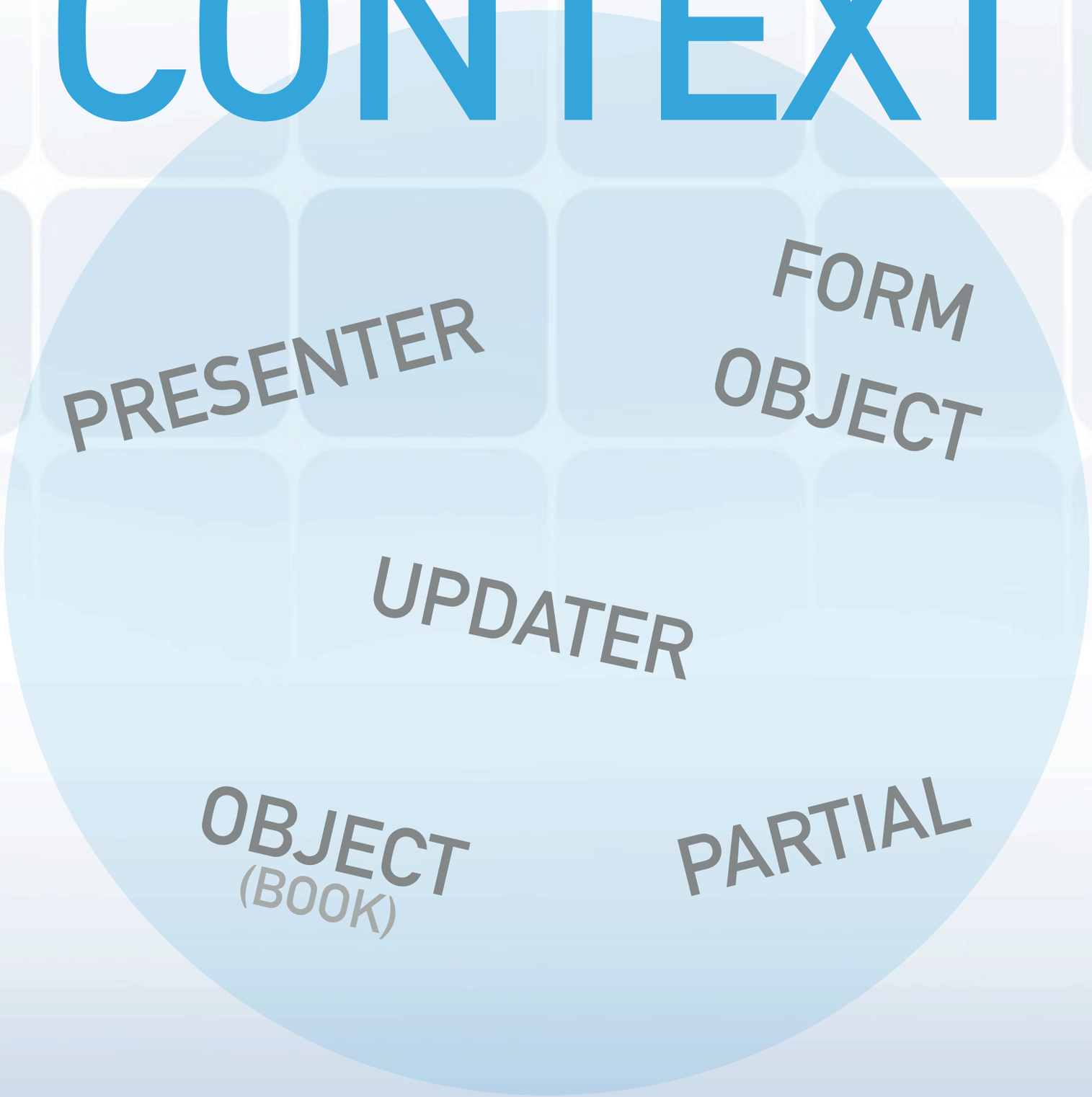
FORM OBJECT

UPDATER

OBJECT (BOOK)

PARTIAL

CONTEXT



CONTEXT

FATtish controllers

PRESENTER

FORM OBJECT

UPDATER

OBJECT (BOOK)

PARTIAL

SERVICES

OBJECTS

SLIMMING THERAPY

CONTEXTS

DIET

context.rb

```ruby
class BookContext

  def initialize(user: nil, params: {})
    @user = user
    @params = params
  end
end

```

```
app
-- services
----- book_updater.rb
----- book_context.rb
```

Line 7, Column 4 Spaces: 2 Ruby

context.rb

```ruby
class BookContext

  def initialize(user: nil, params: {})
    @user = user
    @params = params
  end

  def book
    @book ||= Book.find(params[:id])
  end

  private
  attr_reader :user, :params
end

```

Line 15, Column 1 Spaces: 2 Ruby

context.rb

```ruby
class BookContext

  def initialize(user: nil, params: {})
    @user = user
    @params = params
  end

  def book
    @book ||= Book.find(params[:id])
  end

  def form
    @form ||= form_klass.new(form_params)
  end

  private
  attr_reader :params, :user

  def form_klass
    user.admin? ? ::Books::AdminForm : ::Books::UserForm
  end

  def form_params
    default_params.merge(params[:form_data])
  end
end

```

Line 1, Column 1 Spaces: 2 Ruby

context.rb

```ruby
class BookContext

  def initialize(user: nil, params: {})
    @user = user
    @params = params
  end

  def book
    @book ||= Book.find(params[:id])
  end

  def form
    @form ||= form_klass.new(form_params)
  end

  def updater
    @updater ||= ::BookUpdater.new(book, form)
  end

  private
  attr_reader :params, :user

  def form_klass
    user.admin? ? ::Books::AdminForm : ::Books::UserForm
  end

  def form_params
    default_params.merge(params[:form_data])
```

Line 18, Column 6 Spaces: 2 Ruby

context.rb

```ruby
class BookContext

  def initialize(user: nil, params: {})
    @user = user
    @params = params
  end

  def book
    @book ||= Book.find(params[:id])
  end

  def form
    @form ||= form_klass.new(form_params)
  end

  def updater
    @updater ||= ::BookUpdater.new(book, form)
  end

  def presenter
    @presenter ||= ::BookPresenter.new(book)
  end

  private
  attr_reader :params, :user

  def form_klass
    user.admin? ? ::Books::AdminForm : ::Books::UserForm
```

Line 21, Column 45 Spaces: 2 Ruby

context.rb

```ruby
class BookContext

  def initialize(user: nil, params: {})
    @user = user
    @params = params
  end

  def book
    @book ||= Book.find(params[:id])
  end

  def form
    @form ||= form_klass.new(form_params)
  end

  def updater
    @updater ||= ::BookUpdater.new(book, form)
  end

  def presenter
    @presenter ||= ::BookPresenter.new(book)
  end

  private
  attr_reader :params, :user

  def form_klass
    user.admin? ? ::Books::AdminForm : ::Books::UserForm
```

Line 21, Column 45 Spaces: 2 Ruby

} USER CONTEXT, ECOSYSTEM

books_controller.rb

books_controller.rb

```ruby
class BooksController < ActionController::Base
  #...

  def edit
    @book = Book.find(params[:id])
    @form = build_form(default_params)
    @presenter = ::BookPresenter.new(@book)
    render 'edit'
  end

  def update
    @book = Book.find(params[:id])
    @form = build_form(params)
    @presenter = ::BookPresenter.new(@book)
    if ::BookUpdater.new(@book, @form).perform
      redirect_to book_path(@book)
    else
      if form.valid?
        render 'update_error'
      else
        render 'edit'
      end
    end
  end

  private

  def build_form(form_params)
    current_user.admin? ? ::Books::AdminForm.new(form_params) : ::Books::UserForm.new(form_params)
  end
end

```

31 LINES

Line 1, Column 1 Spaces: 2 Ruby

books_controller.rb

books_controller.rb

```ruby
class BooksController < ActionController::Base
  before_action :load_context
  #...

  def edit; end

  def update
    @context.updater.perform ? redirect_to(book_path(@context.book)) : render(@context.partial_path)
  end

  private

  def load_context
    @context = ::BookContext.new(user: current_user, params: params)
  end
end

```

~~3~~16 LINES

Line 17, Column 1 Spaces: 2 Ruby

books_controller.rb

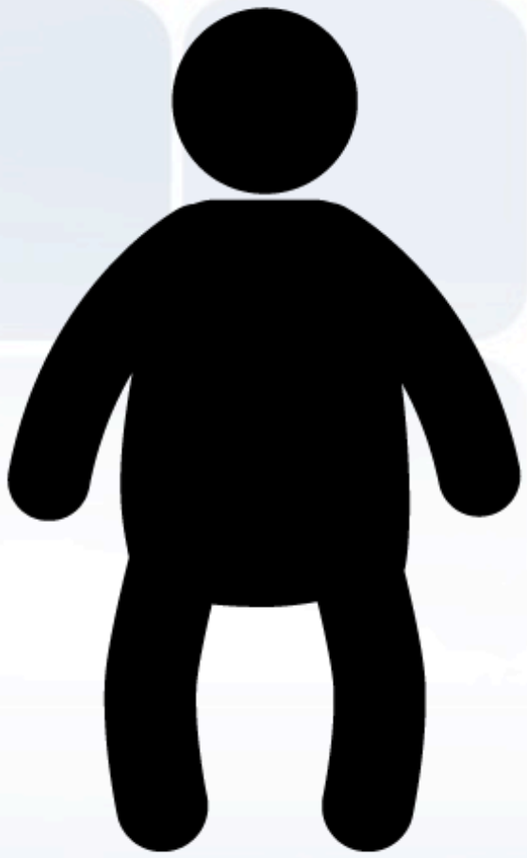
books_controller.rb

```ruby
class BooksController < ActionController::Base
  before_action :load_context
  #...

  def edit; end

  def update
    @context.updater.perform ? redirect_to(book_path(@context.book)) : render(@context.partial_path)
  end

  private

  def load_context
    @context = ::BookContext.new(user: current_user, params: params)
  end
end

```

~~316~~ 316 LINES

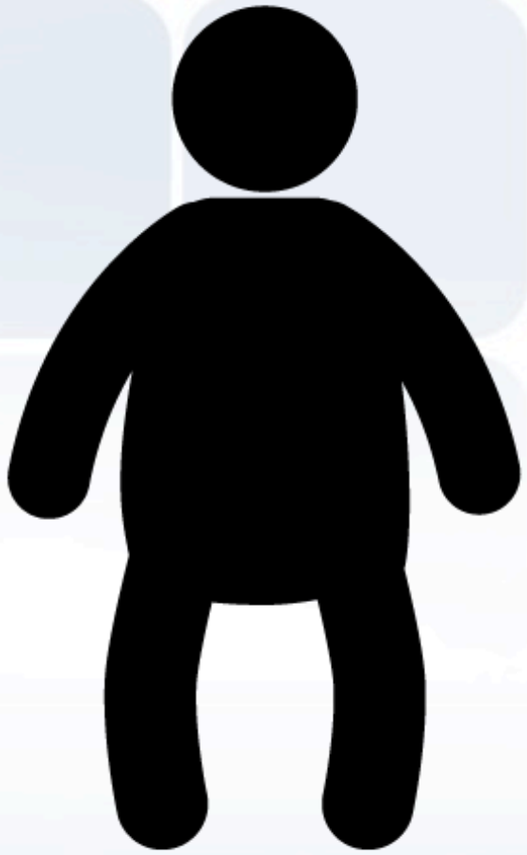
almost 50 %

Line 17, Column 1 Spaces: 2 Ruby



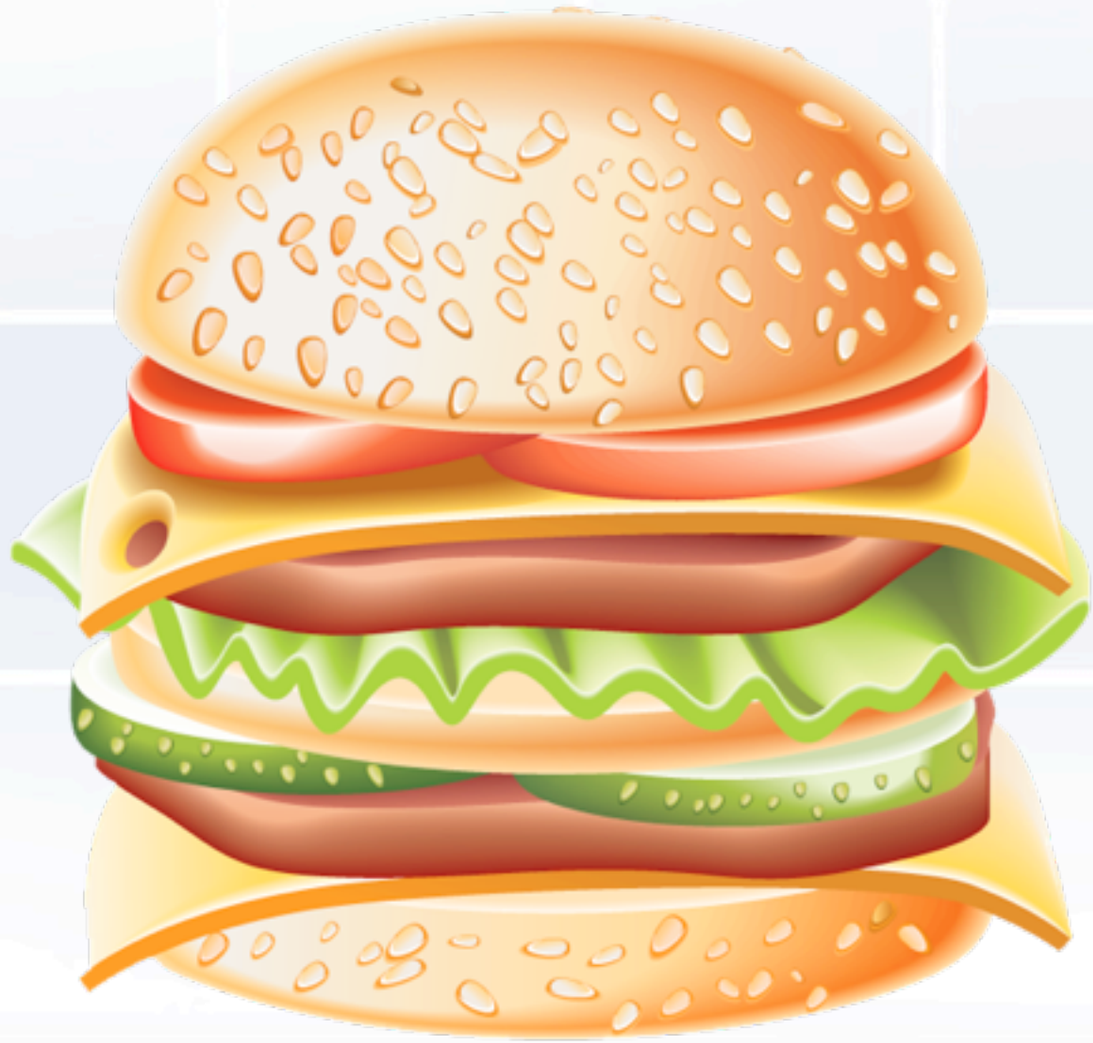
Contexts

diet



Contexts

diet

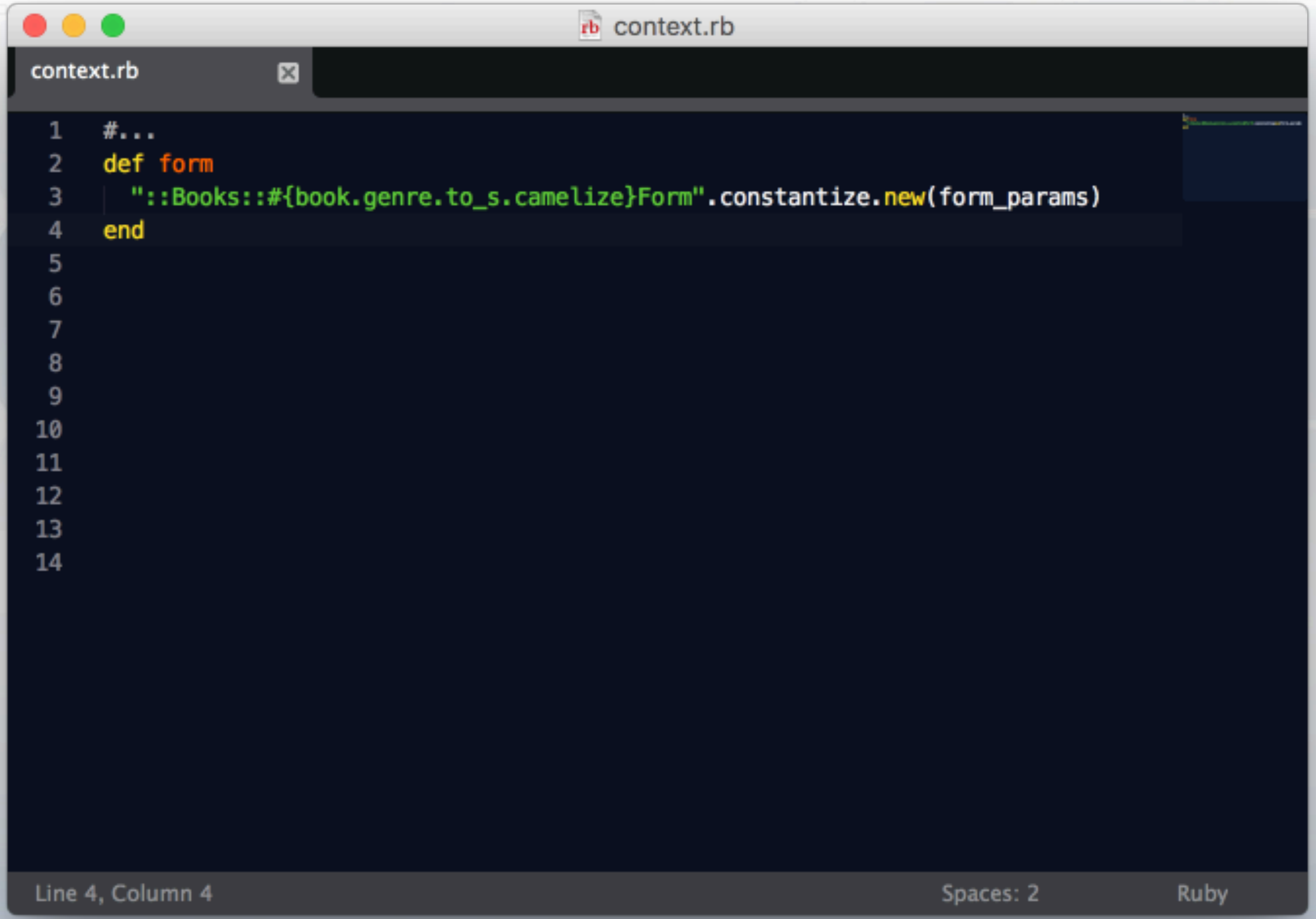


Just a **FAST, slight change**.

5 minutes of work.

FORMULARZ KSIĄŻKI ZALEŻNY OD JEJ GATUNKU


context.rb
context.rb
1 #...
2 def form
3 user.admin? ? ::Books::AdminForm.new(form_params) : ::Books::UserForm.new(
form_params)
4 end
5
Line 5, Column 1
Spaces: 2
Ruby


context.rb
context.rb
#...
def form
"::Books::#{book.genre.to_s.camelize}Form".constantize.new(form_params)
end
Line 4, Column 4
Spaces: 2
Ruby

context.rb

```ruby
#...
def form
  case book.genre
  when :thriller, :horror, :crime, :coaching
    ::Books::FearForm.new(form_params)
  when /fiction/, /histor/
    "::Books::#{book.genre.to_s.camelize}Form".constantize.new(form_params)
  else
    ::Books::BasicForm.new(form_params)
  end
end

```

NIEKTÓRE GATUNKI WSPÓLNY

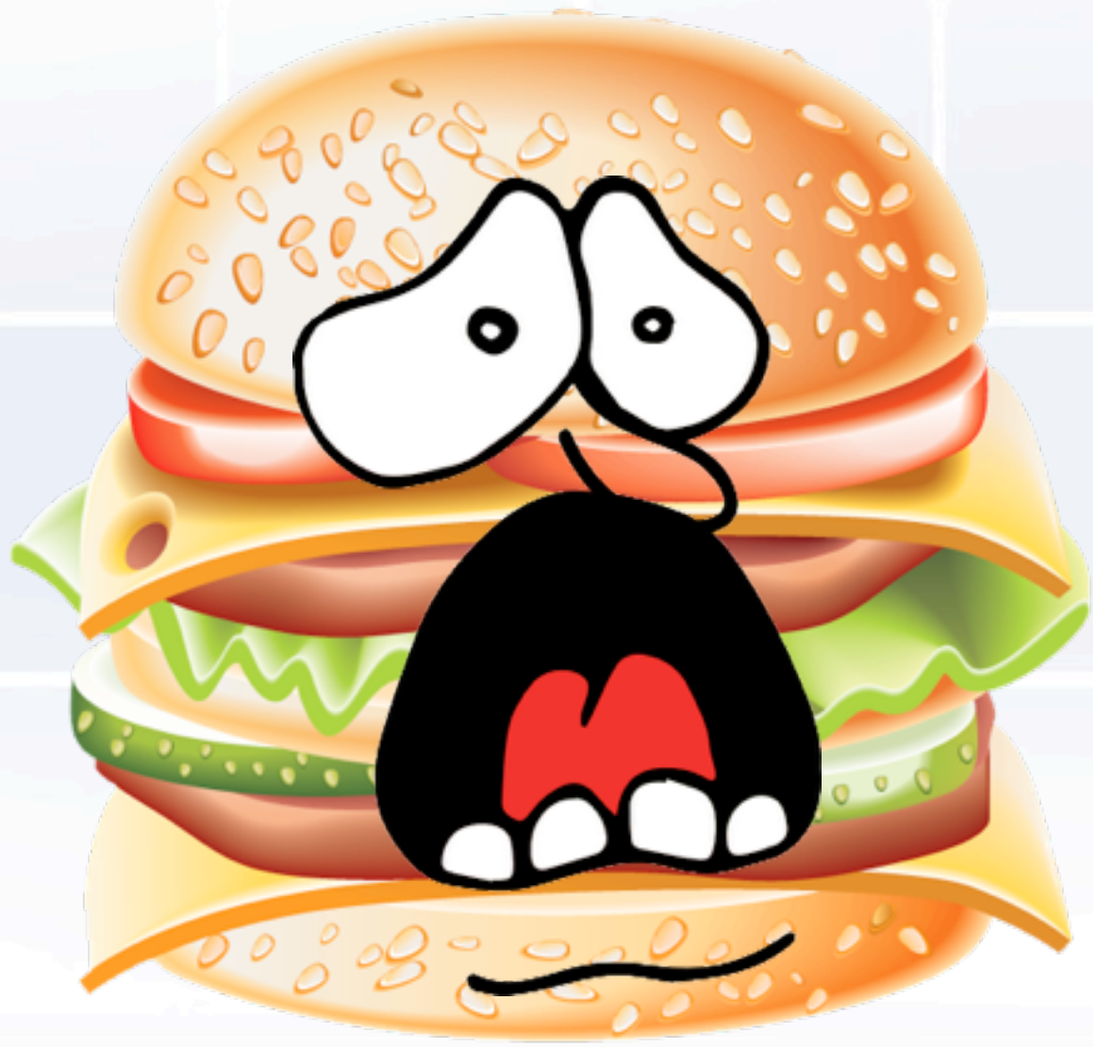
NIEKTÓRE WŁASNY

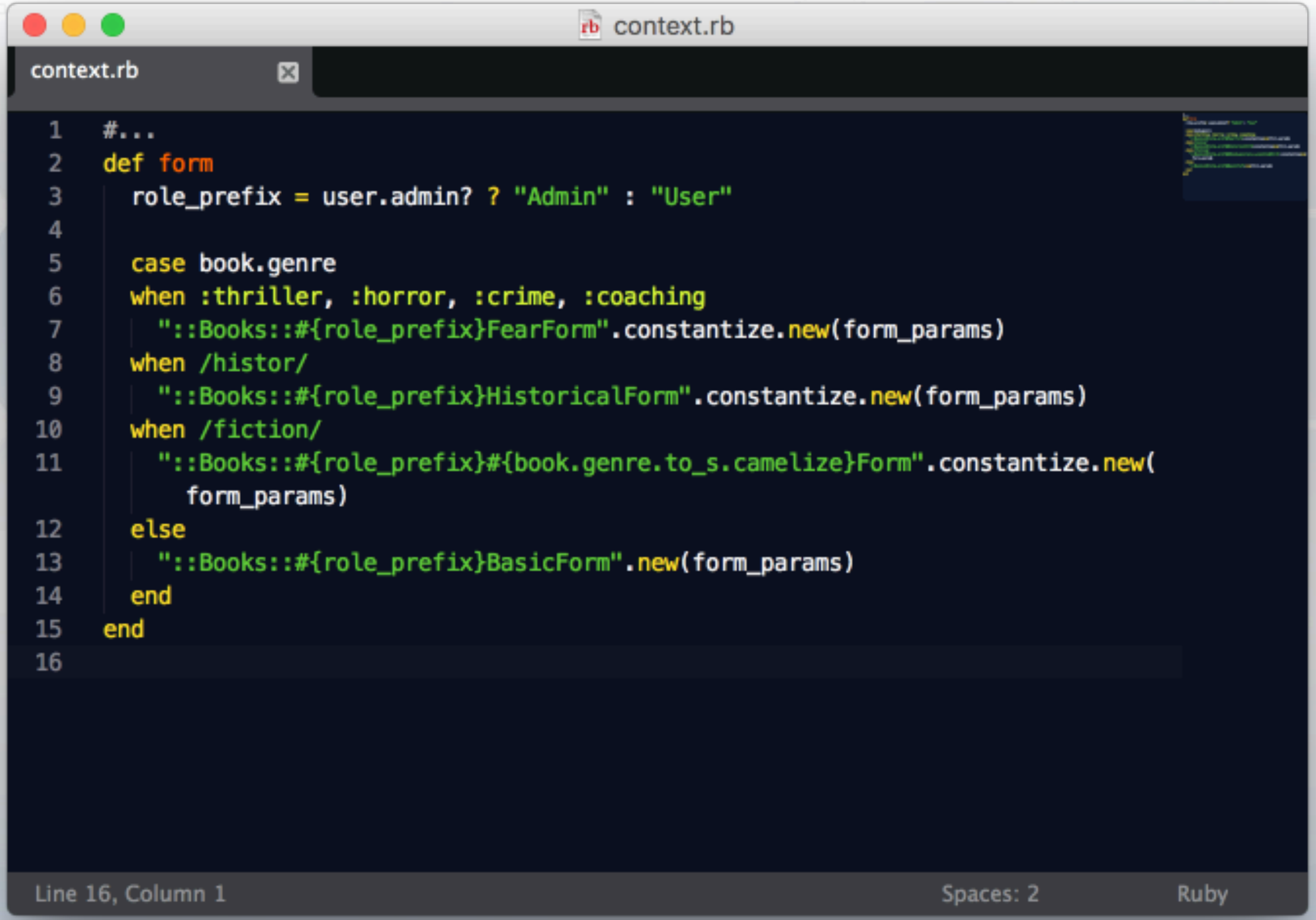
POZOSTAŁE BASIC

Line 1, Column 1 Spaces: 2 Ruby

context.rb

```ruby
#...
def form
  case book.genre
  when :thriller, :horror, :crime, :coaching
    ::Books::FearForm.new(form_params)
  when /histor/
    ::Books::HistoricalForm.new(form_params)
  when /fiction/
    "::Books::#{book.genre.to_s.camelize}Form".constantize.new(form_params)
  else
    ::Books::BasicForm.new(form_params)
  end
end

```




context.rb
context.rb
#...
def form
role_prefix = user.admin? ? "Admin" : "User"
case book.genre
when :thriller, :horror, :crime, :coaching
"::Books::#{role_prefix}FearForm".constantize.new(form_params)
when /histor/
"::Books::#{role_prefix}HistoricalForm".constantize.new(form_params)
when /fiction/
"::Books::#{role_prefix}#{book.genre.to_s.camelize}Form".constantize.new(
form_params)
else
"::Books::#{role_prefix}BasicForm".new(form_params)
end
end
Line 16, Column 1
Spaces: 2
Ruby

context.rb

context.rb

```ruby
class BookContext

  def initialize(user: nil, params: {})
    @user = user
    @params = params
  end

  def book
    @book ||= Book.find(params[:id])
  end

  def form
    role_prefix = user.admin? ? "Admin" : "User"

    @form ||= case book.genre
      when :thriller, :horror, :crime, :coaching
        "::Books::#{role_prefix}FearForm".constantize.new(form_params)
      when /histor/
        "::Books::#{role_prefix}HistoricalForm".constantize.new(form_params)
      when /fiction/
        "::Books::#{role_prefix}#{book.genre.to_s.camelize}Form".constantize.new(form_params)
      else
        "::Books::#{role_prefix}BasicForm".new(form_params)
      end
  end


  def updater
    @updater ||= ::BookUpdater.new(book, form)
  end

  def presenter
    @presenter ||= ::BookPresenter.new(book)
  end

  private
  attr_reader :params, :user
```

Line 25, Column 6 Spaces: 2 Ruby

CONTROLLER…?

UPDATER…?

CONTEXT…?

SKINNY CONTROLLERS.
SKINNY MODELS.
...FAT CONTEXTS?

:book_form,
book,
user

::Book::AdminFearForm

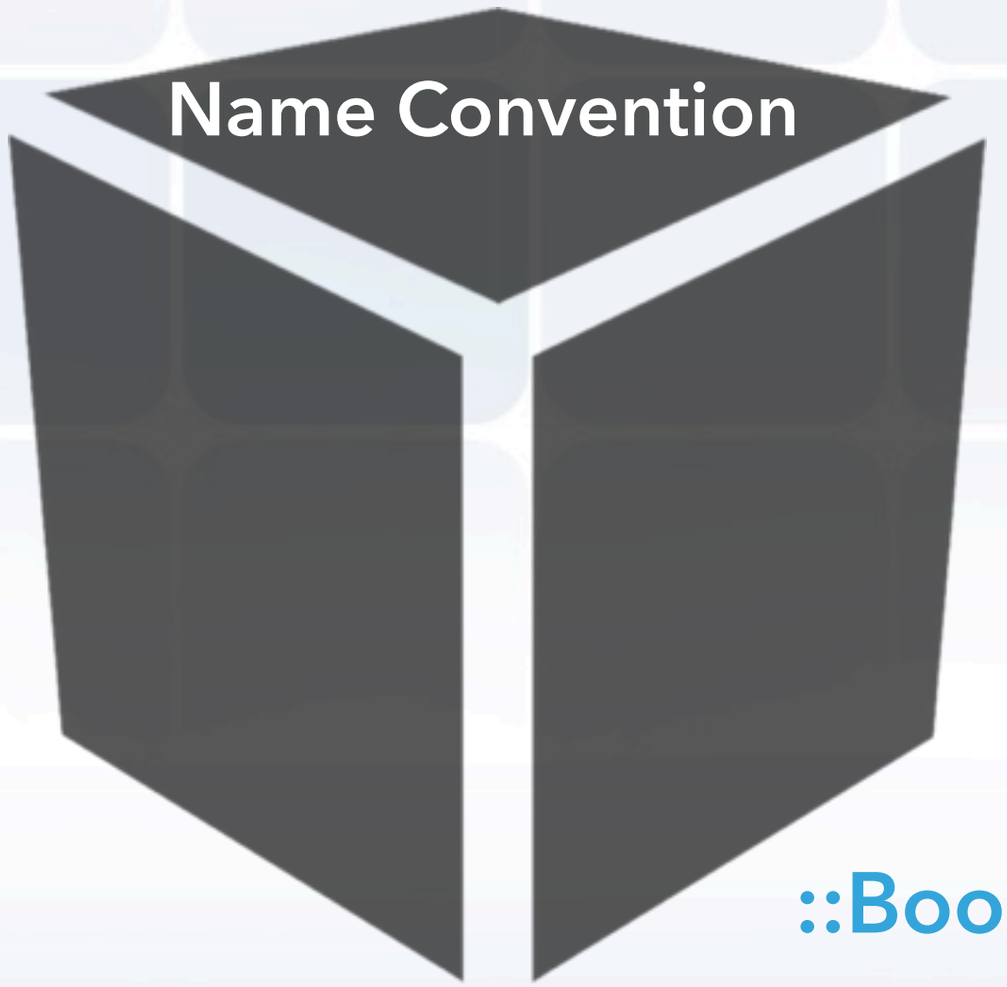
::Book::UserBasicForm

::Book::UserScienceFictionForm

:book_form,

book,

user



::Book::AdminFearForm

::Book::UserBasicForm

::Book::UserScienceFictionForm

SERVICES

OBJECTS

SLIMMING THERAPY

CONTEXTS

DIET

NAME CONVENTION

PERSONAL TRAINER

NAME CONVENTION

DIAGNOZOWANIE KLASY W ZALEŻNOŚCI OD KONTEKSTU

context.rb

```ruby
#...
def form
  role_prefix = user.admin? ? "Admin" : "User"

  case book.genre
  when :thriller, :horror, :crime, :coaching
    "::Books::#{role_prefix}FearForm".constantize.new(form_params)
  when /histor/
    "::Books::#{role_prefix}HistoricalForm".constantize.new(form_params)
  when /fiction/
    "::Books::#{role_prefix}#{book.genre.to_s.camelize}Form".constantize.new(
      form_params)
  else
    "::Books::#{role_prefix}BasicForm".new(form_params)
  end
end

```

Line 16, Column 1 Spaces: 2 Ruby

book_form.rb

book_form.rb

```ruby
module NameConvention
  class BookForm
    def initialize(user, book)
      @user = user
      @book = book
    end

    def klass_name
      "::Books::#{role_prefix}#{klass_sufix}Form".constantize
    end

    private
    attr_reader :user, :book

    def role_prefix
      @role_prefix ||= user.admin? ? "Admin" : "User"
    end

    def klass_sufix
      case book.genre
      when :thriller, :horror, :crime, :coaching then "Fear"
      when /histor/ then "Historical"
      when /fiction/ then book.genre.to_s.camelize
      else "Basic"
      end
    end
  end
end
```

Line 9, Column 62 Spaces: 2 Ruby

app

—— services

———— name_convention

——————— book_form.rb

context.rb

context.rb

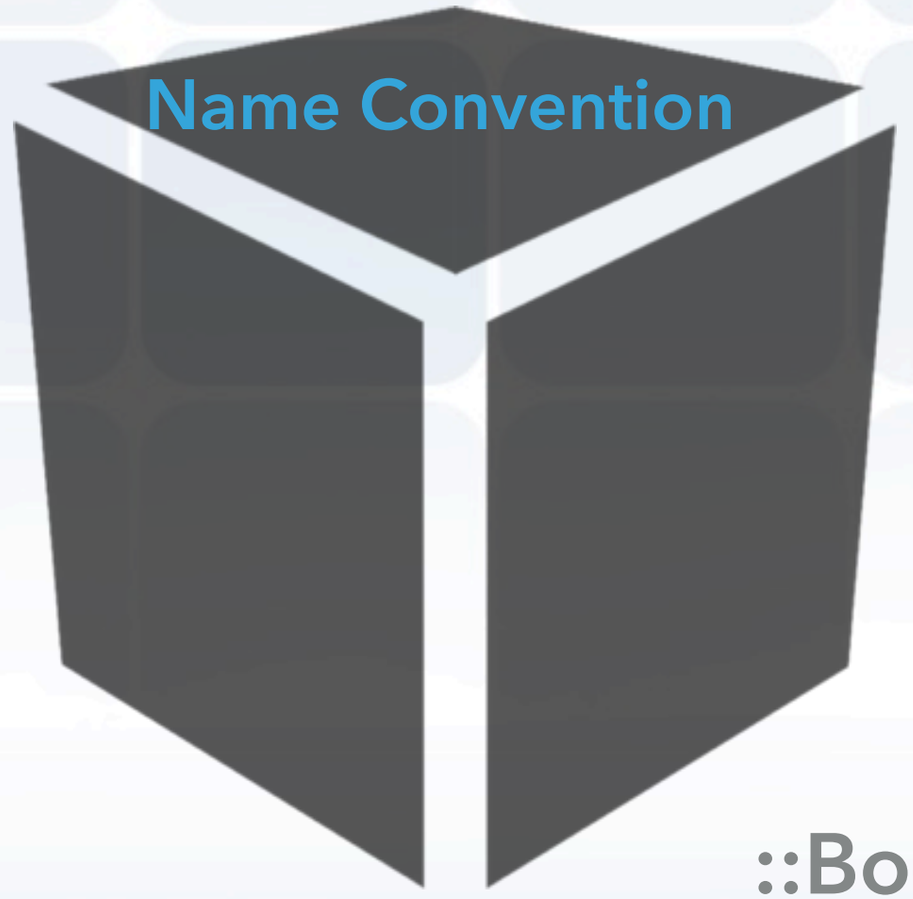
```ruby
#...

  def form
    @form ||= form_klass.new(form_params)
  end

  def form_klass
    ::NameConvention::BookForm.new(user, book).klass_name
  end

#...

```

Line 12, Column 1 Spaces: 2 Ruby

:book_form,

book,

user



::Book::AdminFearForm

::Book::UserBasicForm

::Book::UserScienceFictionForm

NameConvention.klass_name(:book_form, user: user, book: book)

NameConvention.klass_name(:book_updater, book)

name_convention.rb

name_convention.rb

```ruby
module NameConvention
  module_function

  def identify
  end

  def klass_name
  end
end

```

app

—— services

———— book_updater.rb

———— book_context.rb

———— name_convention.rb

Line 10, Column 1 Spaces: 2 Ruby

name_convention.rb

name_convention.rb

```ruby
module NameConvention
  module_function

  def identify(name, object)
  end

  def klass_name(*args)
    identify(*args).to_s.constantize
  end
end

```

NameConvention.klass_name(:book_form, user: user, book: book)

Line 8, Column 37 Spaces: 2 Ruby

name_convention.rb

name_convention.rb

```ruby
module NameConvention
  module_function

  def identify(name, object)
  end

  def klass_name(*args)
    identify(*args).to_s.constantize
  end
end

```

NameConvention.klass_name(:book_form, user: user, book: book)

Line 8, Column 17 Spaces: 2 Ruby

name_convention.rb

name_convention.rb

```ruby
module NameConvention
  module_function

  def identify(name, object)
    "NameConvention::#{name.to_s.camelize}"
  end

  def klass_name(*args)
    identify(*args).to_s.constantize
  end
end

```

NameConvention.klass_name(:book_form, user: user, book: book)

Line 5, Column 44 Spaces: 2 Ruby

name_convention.rb

name_convention.rb

```ruby
module NameConvention
  module_function

  def identify(name, object)
    "NameConvention::#{name.to_s.camelize}"
  end

  def klass_name(*args)
    identify(*args).to_s.constantize
  end
end

```

"NameConvention::BookForm"

NameConvention.klass_name(:book_form, user: user, book: book)

Line 5, Column 44 Spaces: 2 Ruby

name_convention.rb

name_convention.rb

```ruby
module NameConvention
  module_function

  def identify(name, object)
    "NameConvention::#{name.to_s.camelize}".constantize.new(object)
  end

  def klass_name(*args)
    identify(*args).to_s.constantize
  end
end
```

user: user, book: book

NameConvention.klass_name(:book_form, user: user, book: book)

Line 5, Column 68 Spaces: 2 Ruby

name_convention.rb

name_convention.rb

```ruby
module NameConvention
  module_function

  def identify(name, object)
    "NameConvention::#{name.to_s.camelize}".constantize.new(object).name
  end

  def klass_name(*args)
    identify(*args).to_s.constantize
  end
end

```

NameConvention.klass_name(:book_form, user: user, book: book)

Line 5, Column 73 Spaces: 2 Ruby

book_form.rb

```ruby
module NameConvention
  class BookForm
    def initialize(user, book)
      @user = user
      @book = book
    end

    def klass_name
      "::Books::#{role_prefix}#{klass_sufix}Form".constantize
    end

    private
    attr_reader :user, :book

    def role_prefix
      @role_prefix ||= user.admin? ? "Admin" : "User"
    end

    def klass_sufix
      case book.genre
      when :thriller, :horror, :crime, :coaching then "Fear"
      when /histor/ then "Historical"
      when /fiction/ then book.genre.to_s.camelize
      else "Basic"
      end
    end
  end
end
```

Line 9, Column 62 Spaces: 2 Ruby

book_form.rb

```ruby
module NameConvention
  class BookForm
    def initialize(user: nil, book: nil)
      @user = user
      @book = book
    end

    def name
      "::Books::#{role_prefix}#{klass_sufix}Form"
    end

    private
    attr_reader :user, :book

    def role_prefix
      @role_prefix ||= user.admin? ? "Admin" : "User"
    end

    def klass_sufix
      case book.genre
      when :thriller, :horror, :crime, :coaching then "Fear"
      when /histor/ then "Historical"
      when /fiction/ then book.genre.to_s.camelize
      else "Basic"
      end
    end
  end
end
```

Line 9, Column 50 Spaces: 2 Ruby

context.rb

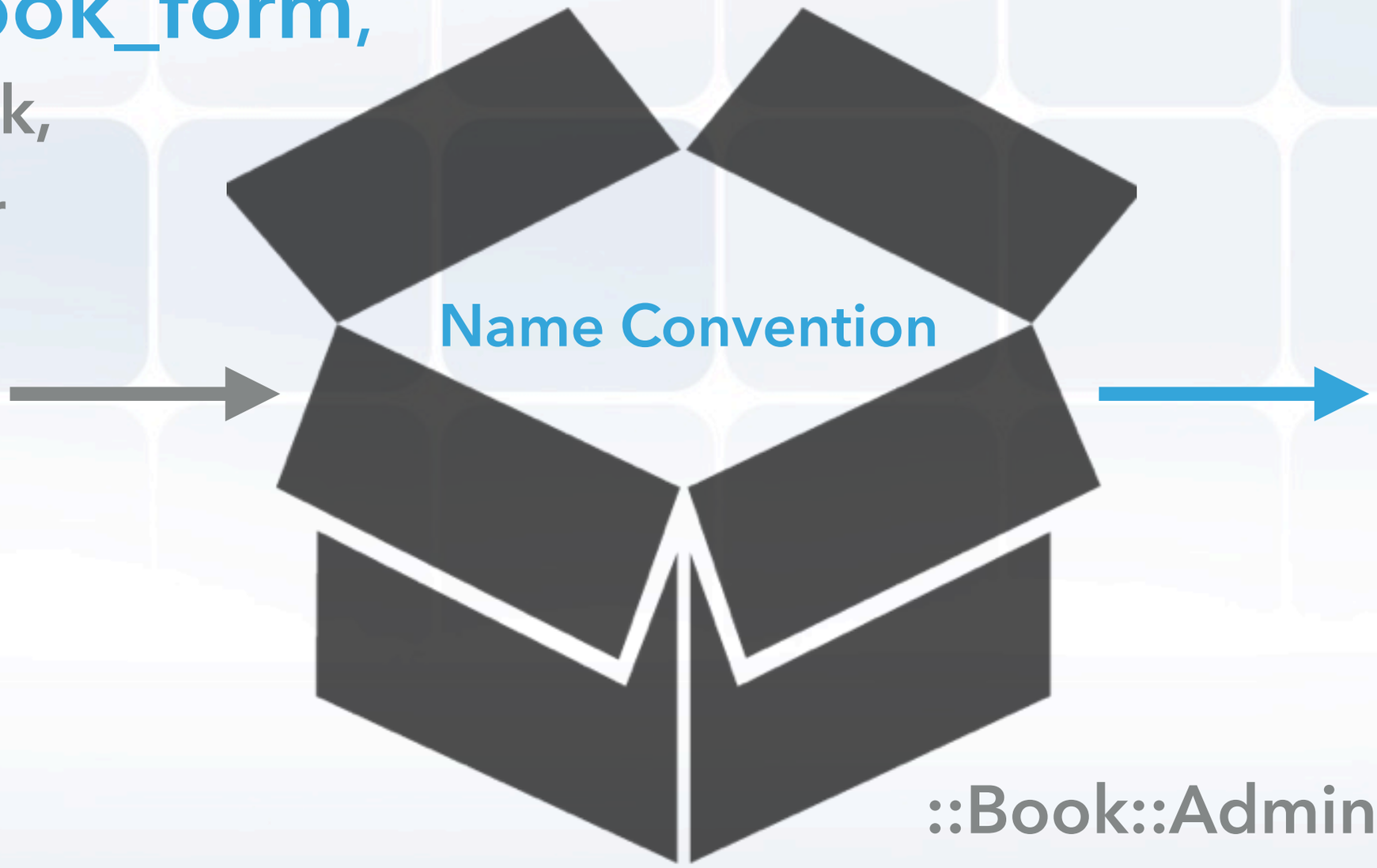
```ruby
#...

  def form
    @form ||= form_klass.new(form_params)
  end

  def form_klass
    ::NameConvention.klass_name(:book_form, user: user, book: book)
  end

#...

```

Line 12, Column 1 Spaces: 2 Ruby

:book_form,

book,

user



::Book::AdminFearForm

::Book::UserBasicForm

::Book::UserScienceFictionForm

Name Convention

Personal Trainer


Celebrity SOLSTICE

SERVICES OBJECTS

(SKOMPLIKOWANA) LOGIKA BIZNESOWA PRZENIESIONA DO SERWISÓW

CONTEXTS

ELEMENTY TWORZĄCE KONTEKST WYWOŁANIA - DO SERWISÓW KONTEKSTOWYCH

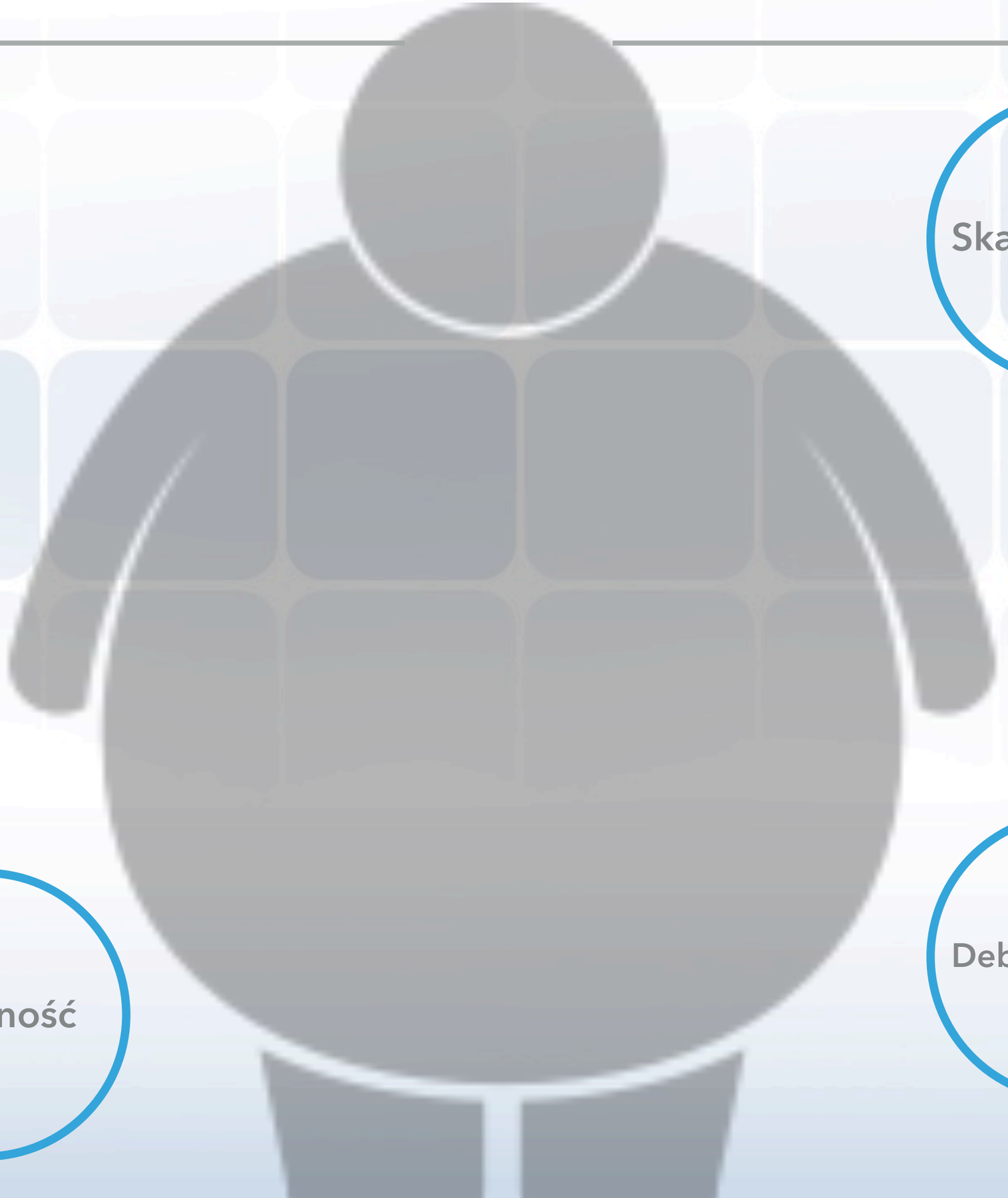
NAME CONVENTION

STWORZONY MECHANIZM
ODCIĄŻAJĄCY KOD
Z DIAGNOZOWANIA KLASY

Choroby serca

Problemy z oddychaniem

Zwyrodnienia stawów



Skalowalność

Maintainability

Stabilność

Debuggowanie

MAKE YOUR CODE FIT AGAIN

Alicja Cyganiewicz

alicja.cyganiewicz@infakt.pl

KRUG #6 / 2017

powered by Railsware.pl

17.10.2017 r.

MAKE YOUR CODE FIT AGAIN

Warsztaty
Ruby on Rails
z inFakt

www.infakt.pl/warsztaty-rails

www.meetup.com/Warsztaty-Ruby-on-Rails-z-inFakt/

FOR FREE

8, 15, 22 listopada

junior+

5 mentorów