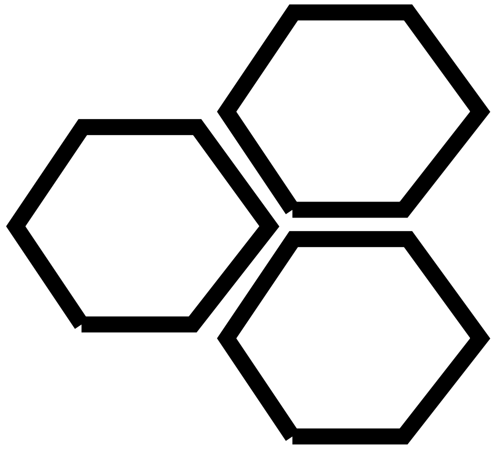
Refactoring Workshop

Hexagonal Rails

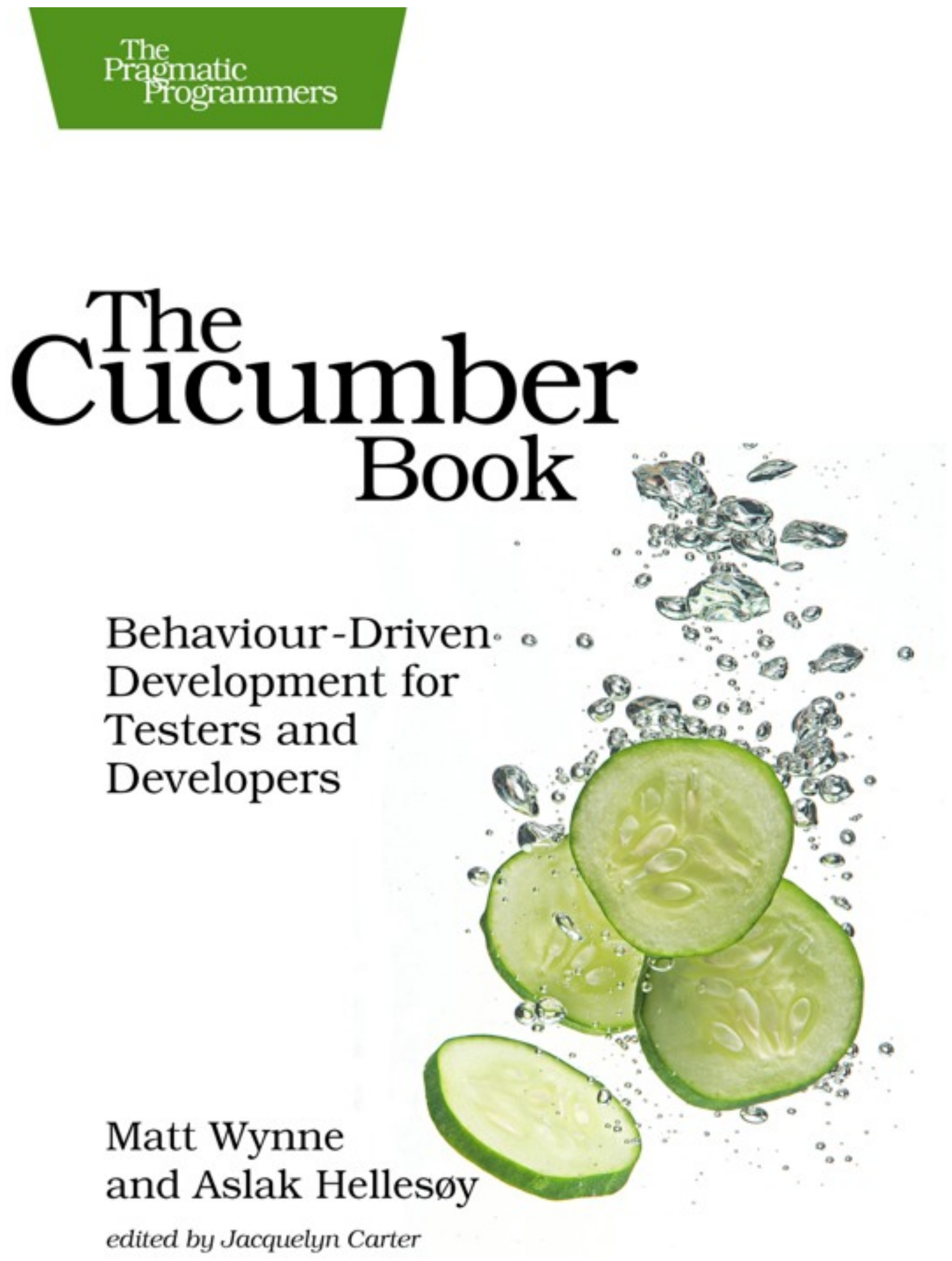


@mattwynne

GOTO Amsterdam | 18th June 2013

Structure of this session:

- Installing things (first pass)
- Intro presentation
- Installing things (for real)
- Refactoring exercise
- Further exploration
- Show & tell


The Pragmatic Programmers

The Cucumber Book

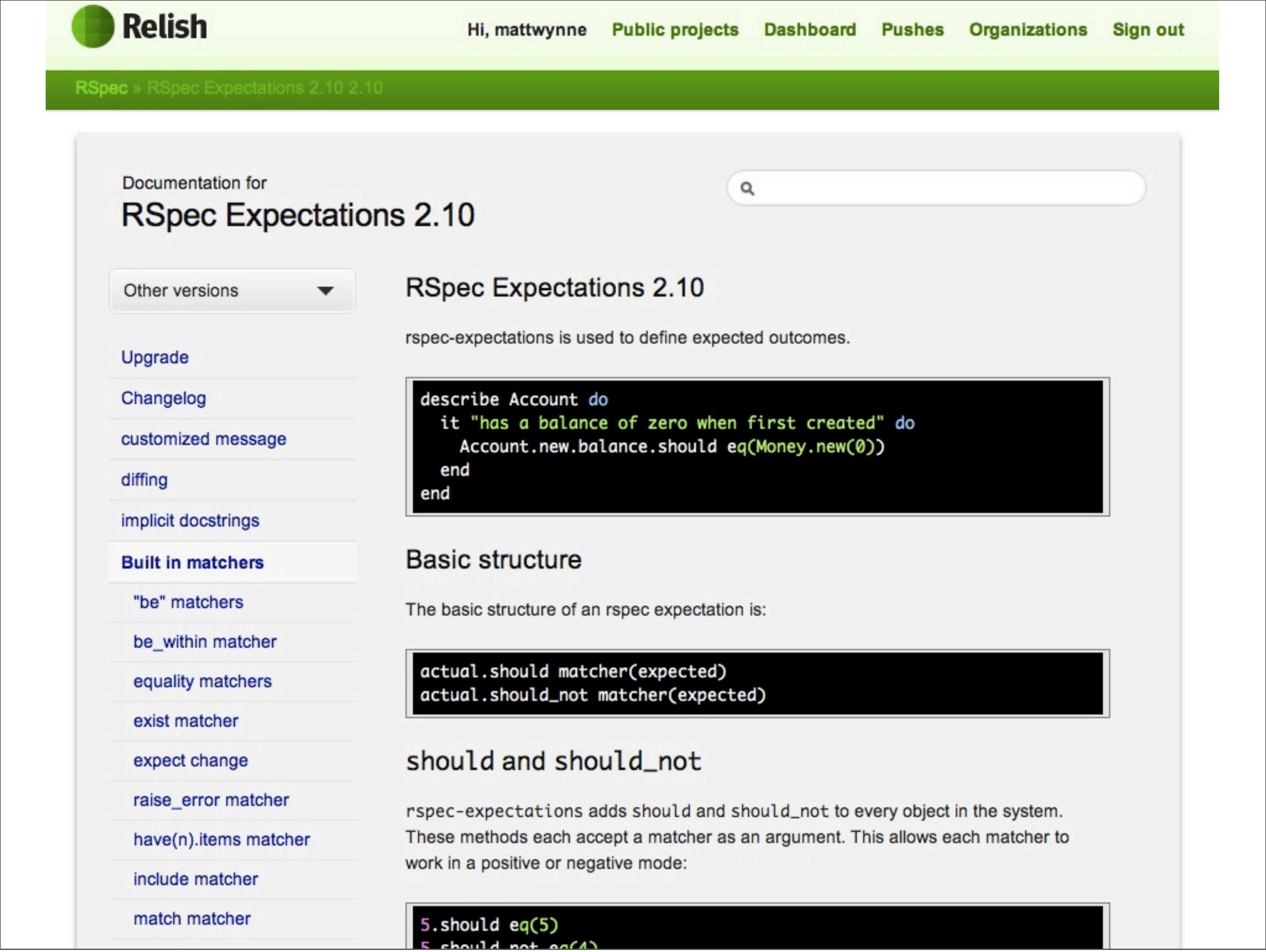
Behaviour-Driven Development for Testers and Developers

Matt Wynne and Aslak Hellesøy

edited by Jacquelyn Carter

Relish

Living documentation



Documentation for

RSpec Expectations 2.10

RSpec Expectations 2.10

rspec-expectations is used to define expected outcomes.

```
describe Account do
  it "has a balance of zero when first created" do
    Account.new.balance.should eq(Money.new(0))
  end
end
```

Basic structure

The basic structure of an rspec expectation is:

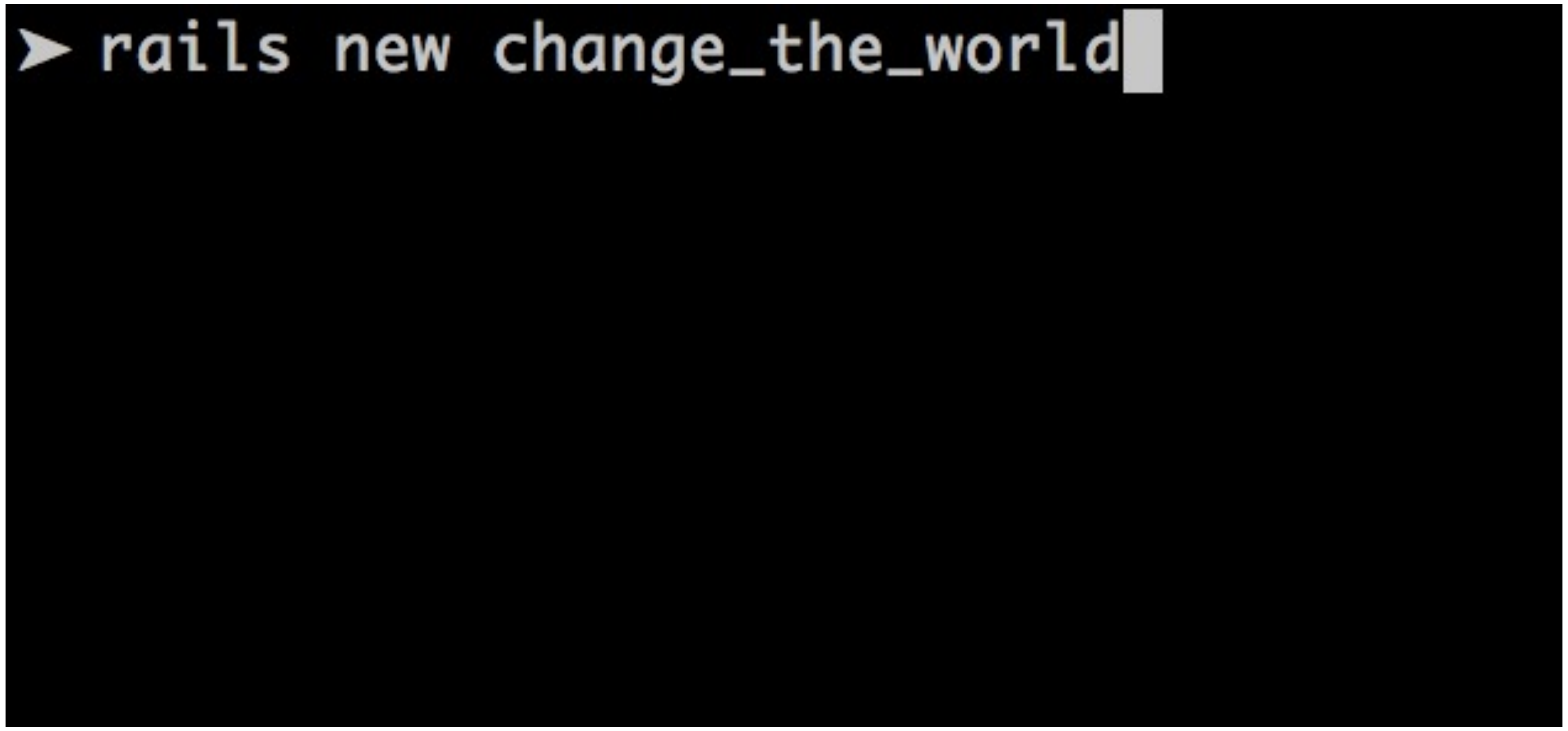
```
actual.should matcher(expected)
actual.should_not matcher(expected)
```

should and should_not

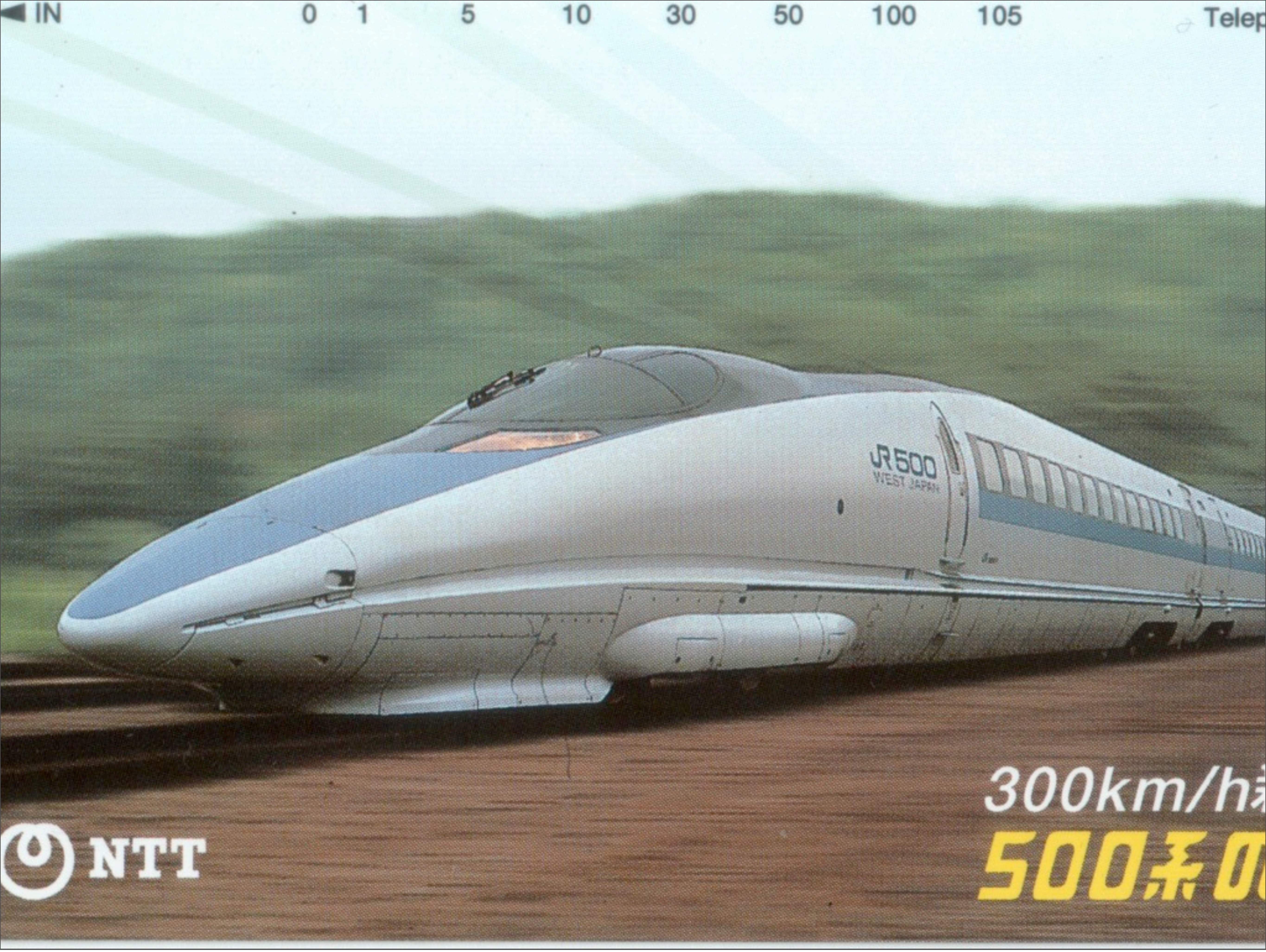
`rspec-expectations` adds `should` and `should_not` to every object in the system. These methods each accept a matcher as an argument. This allows each matcher to work in a positive or negative mode:

```
5.should eq(5)
```

Tuesday, 18 June 13



```
➤ rails new change_the_world
```

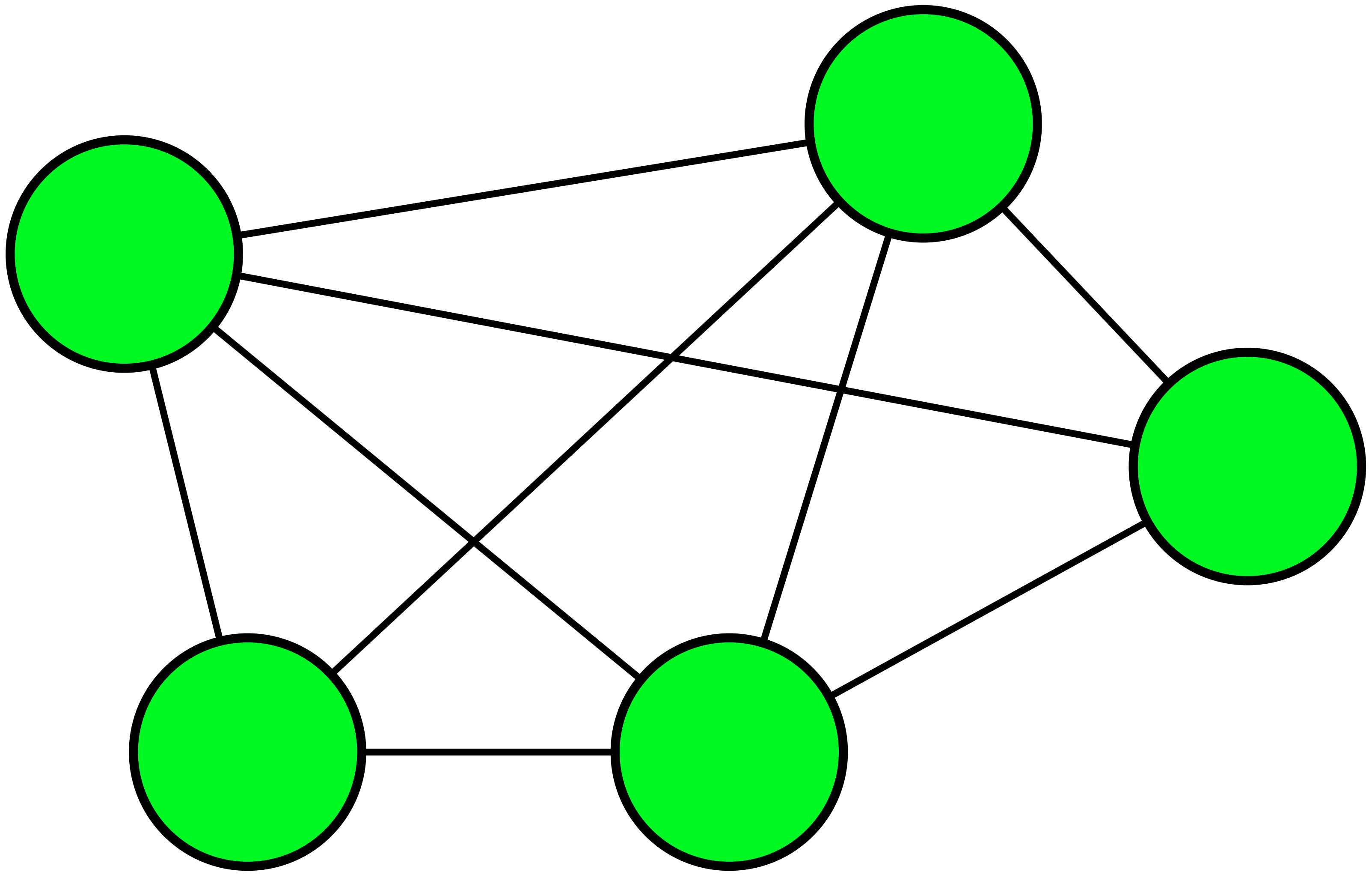

◀ IN
0 1 5 10 30 50 100 105
Telep
JR500
WEST JAPAN
300km/h
500系0
NTT

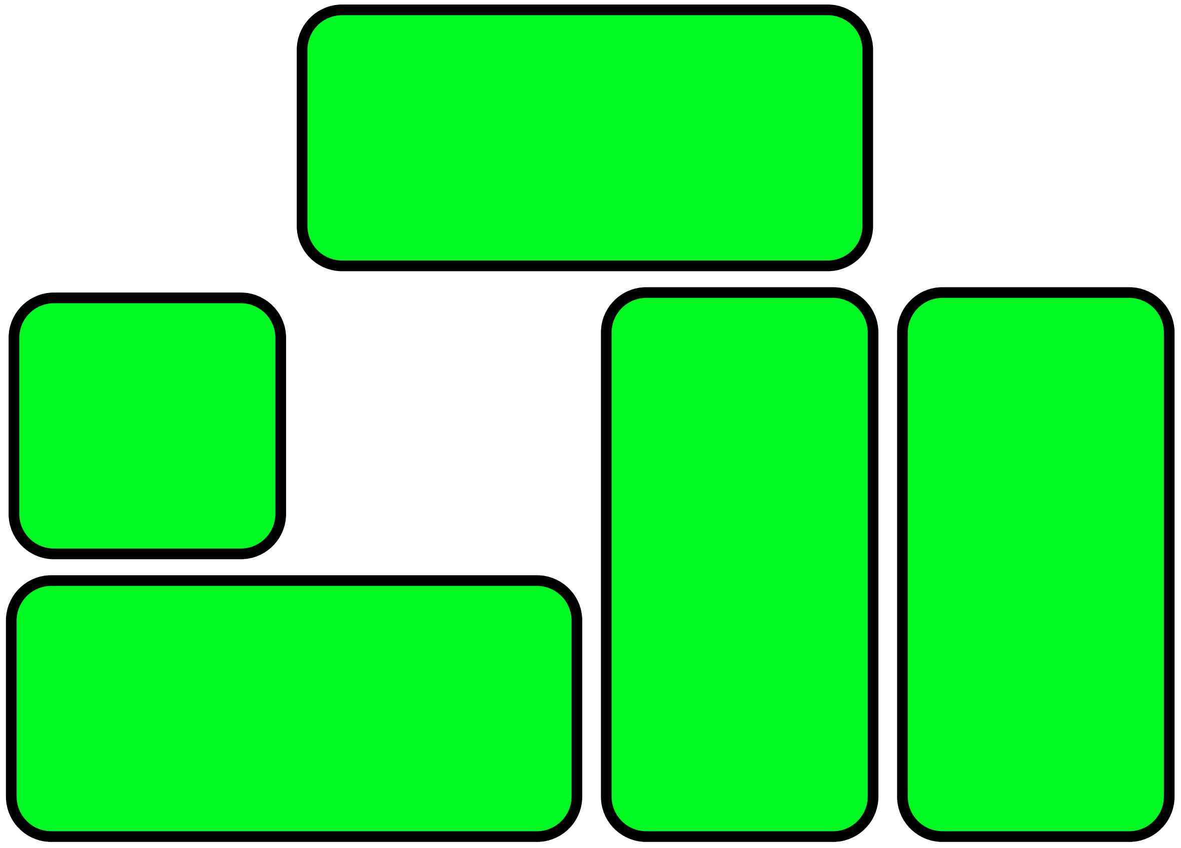
and then...

Why?

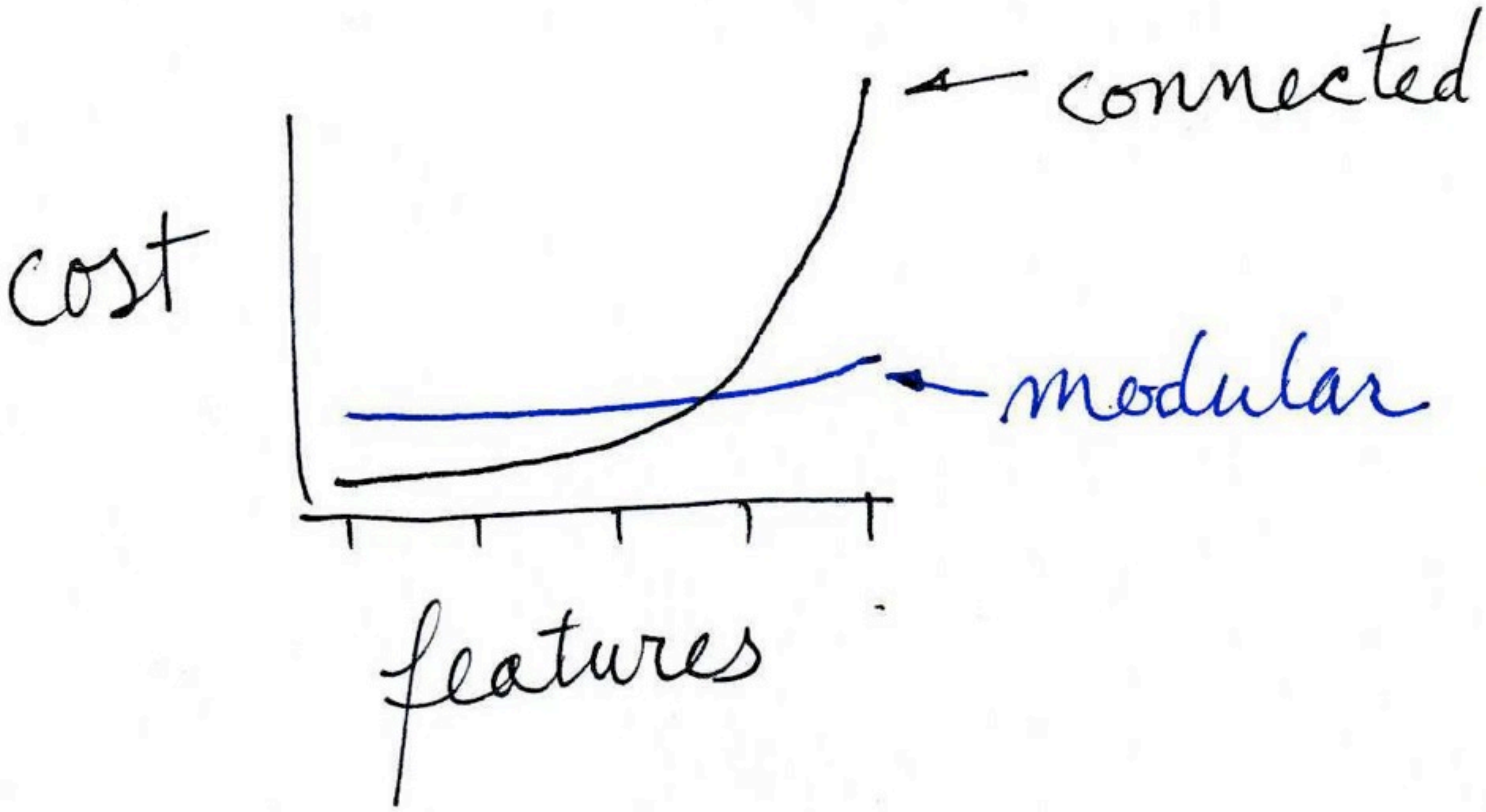
it-agile

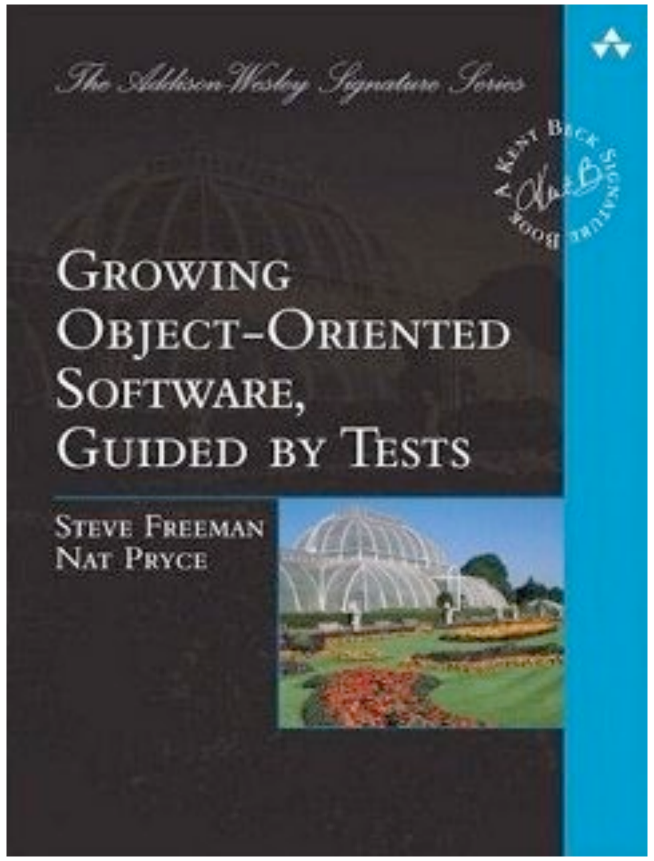
Connected



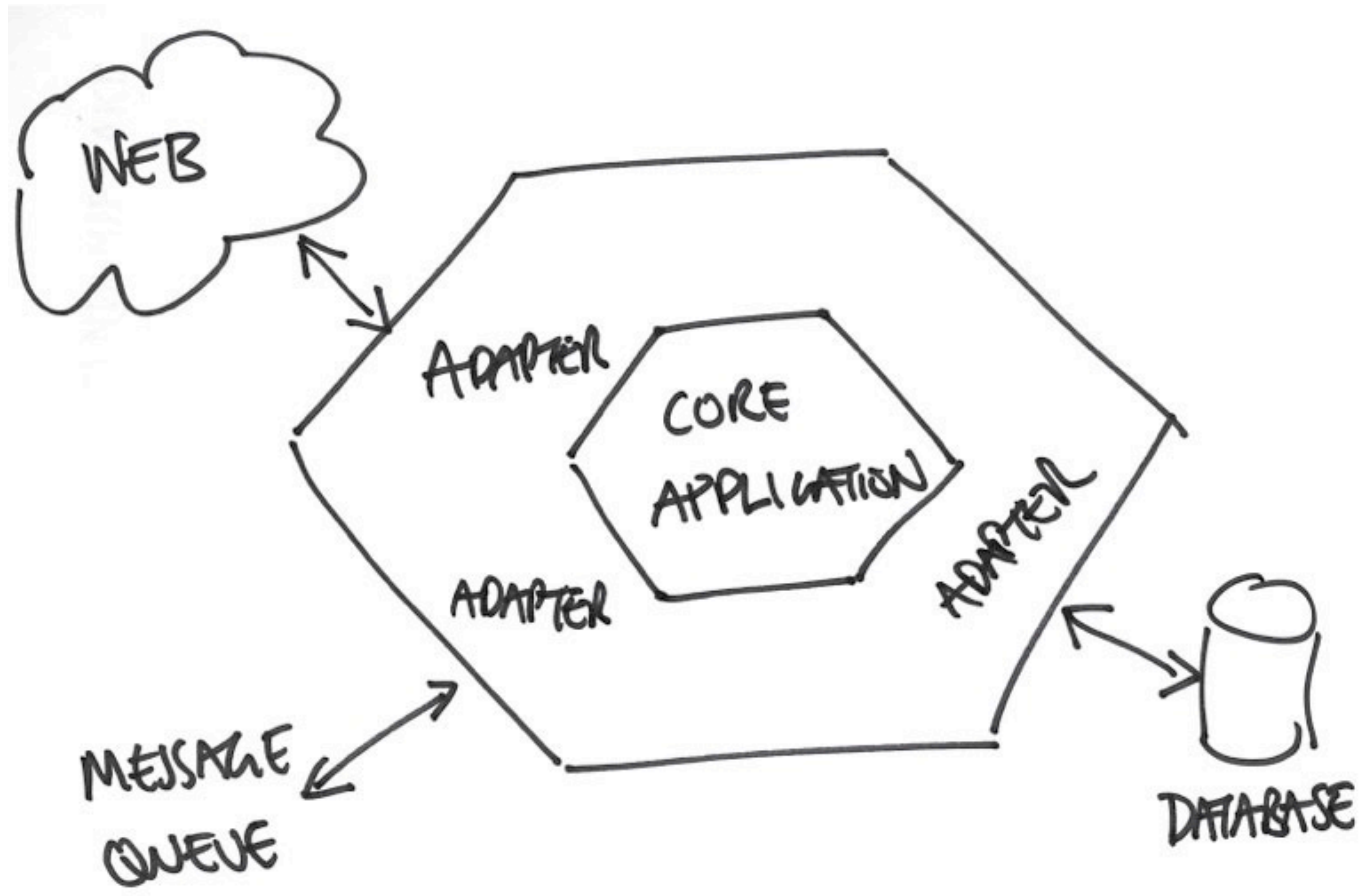


Modular


connected
cost
modular
features


The Addison-Wesley Signature Series
A Kent Beck Signature Book
Growing Object-Oriented Software, Guided by Tests
Steve Freeman
Nat Pryce

Modular, in the large


WEB
ADAPTER
CORE APPLICATION
ADAPTER
ADAPTER
MESSAGE QUEUE
DATABASE

Modular,
in the small

Your domain model is not in your classes, it's ***in the communication patterns between the objects at runtime****.*

Your domain model is not in your classes, it's ***in the communication patterns between the objects at runtime***.

— Steve Freeman & Nat Price

Your domain model is not where you think it is

Your domain model is in the protocols

Procedural code gets information then makes decisions. Object-oriented code tells objects to do things.

— Alec Sharp, Smalltalk by Example

Procedural code gets information then makes decisions. Object-oriented code tells objects to do things.

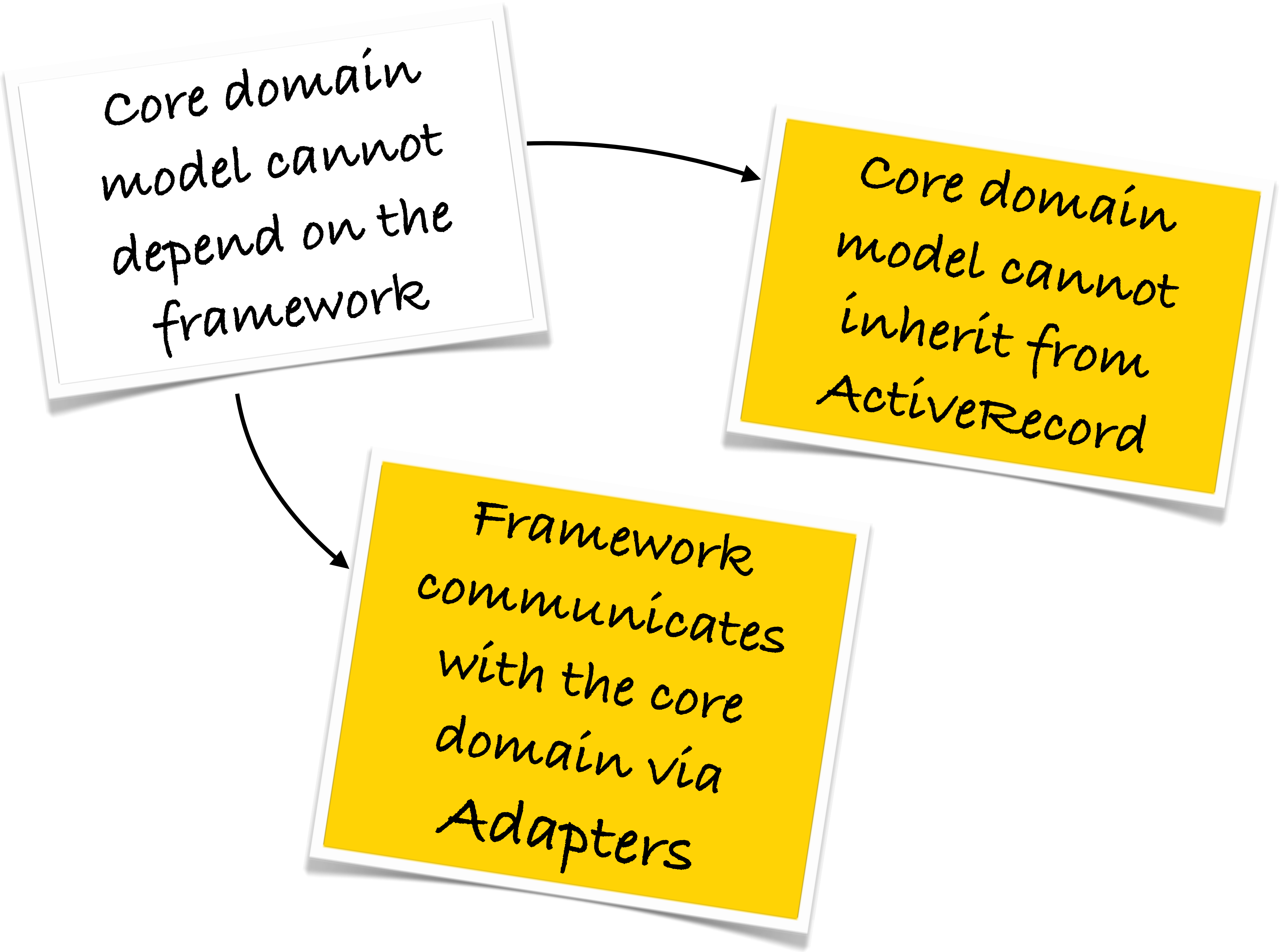
— Alec Sharp, Smalltalk by Example

Core domain model cannot depend on the framework

Rules

Tell objects, ask values

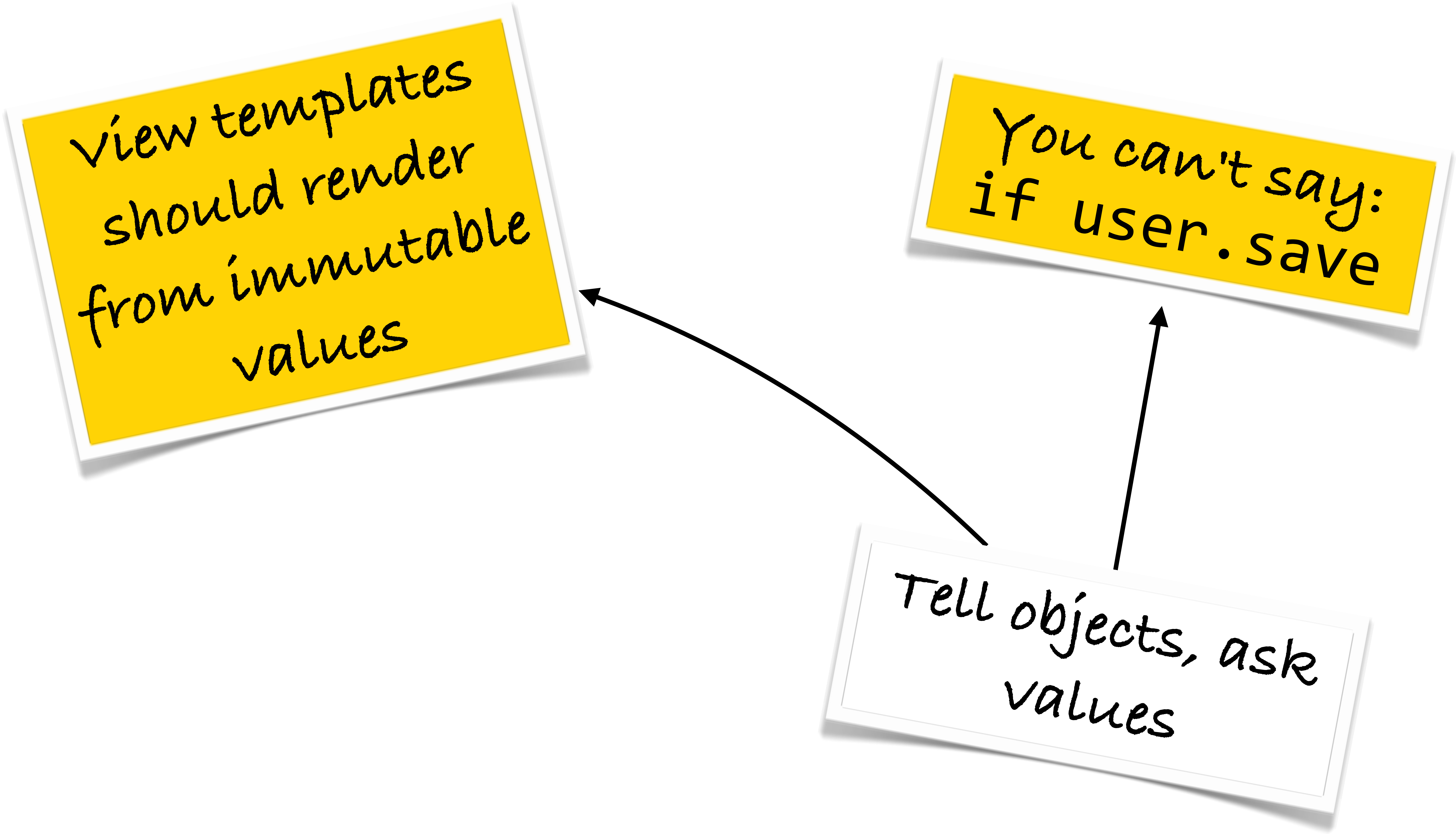
Implications



Core domain model cannot depend on the framework

Core domain model cannot inherit from ActiveRecord

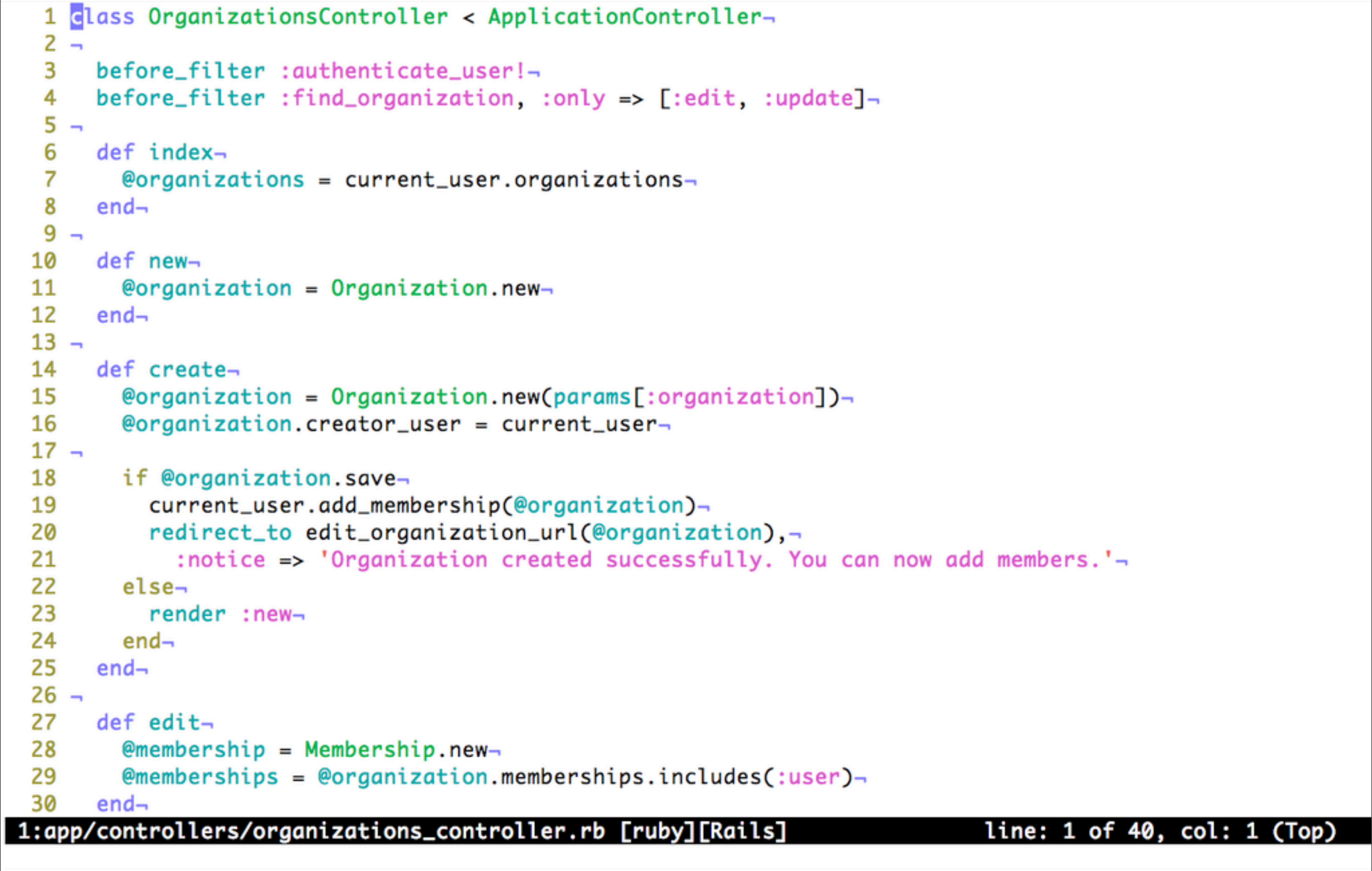
Framework communicates with the core domain via Adapters


View templates should render from immutable values

You can't say: `if user.save`

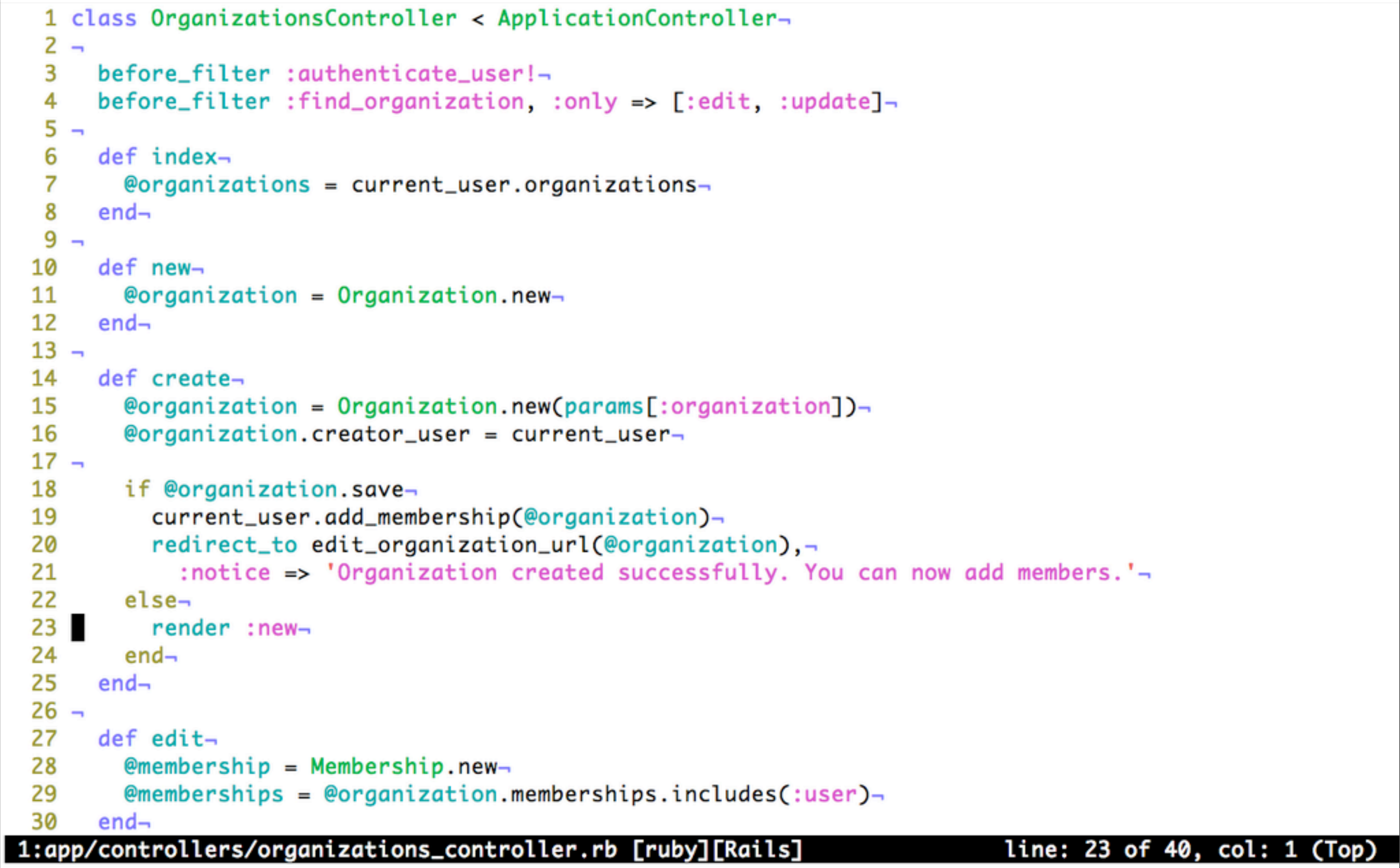
Tell objects, ask values

Example:
Passive Controller



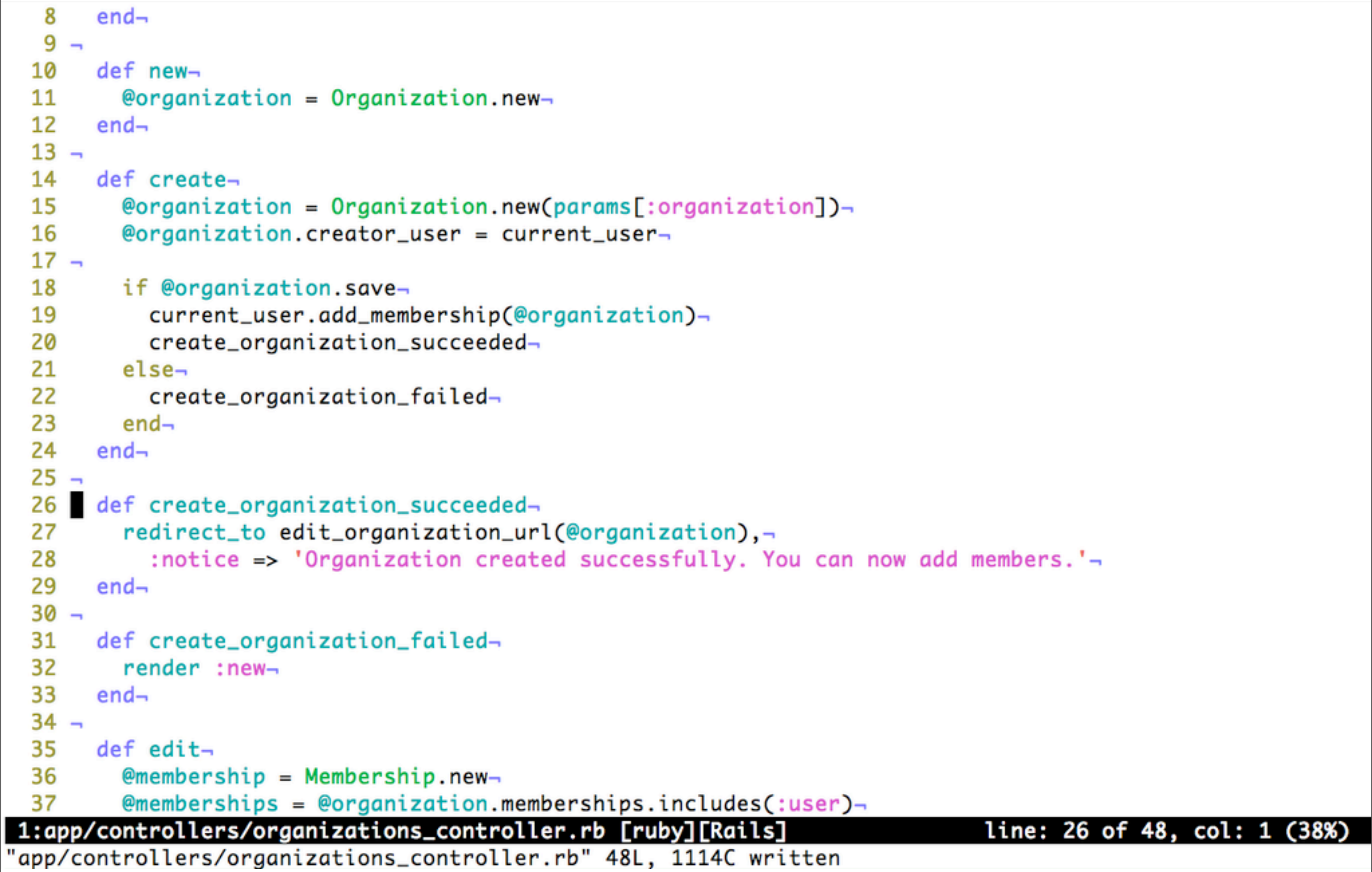
```ruby
class OrganizationsController < ApplicationController

  before_filter :authenticate_user!
  before_filter :find_organization, :only => [:edit, :update]

  def index
    @organizations = current_user.organizations
  end

  def new
    @organization = Organization.new
  end

  def create
    @organization = Organization.new(params[:organization])
    @organization.creator_user = current_user

    if @organization.save
      current_user.add_membership(@organization)
      redirect_to edit_organization_url(@organization),
        :notice => 'Organization created successfully. You can now add members.'
    else
      render :new
    end
  end

  def edit
    @membership = Membership.new
    @memberships = @organization.memberships.includes(:user)
  end
```

```
1:app/controllers/organizations_controller.rb [ruby][Rails]          line: 1 of 40, col: 1 (Top)
```

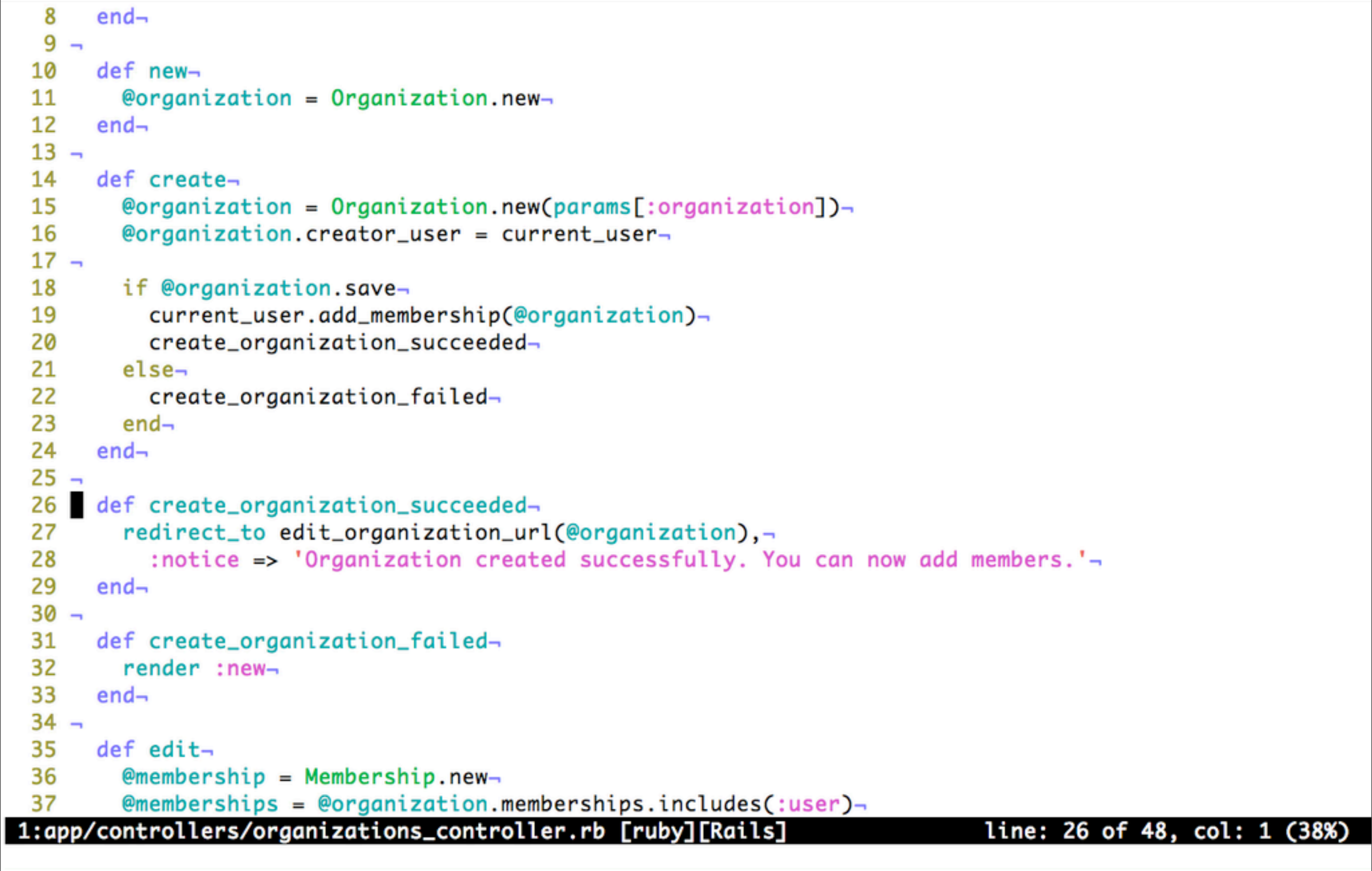


```ruby
class OrganizationsController < ApplicationController

  before_filter :authenticate_user!
  before_filter :find_organization, :only => [:edit, :update]

  def index
    @organizations = current_user.organizations
  end

  def new
    @organization = Organization.new
  end

  def create
    @organization = Organization.new(params[:organization])
    @organization.creator_user = current_user

    if @organization.save
      current_user.add_membership(@organization)
      redirect_to edit_organization_url(@organization),
        :notice => 'Organization created successfully. You can now add members.'
    else
      render :new
    end
  end

  def edit
    @membership = Membership.new
    @memberships = @organization.memberships.includes(:user)
  end
```

```
1:app/controllers/organizations_controller.rb [ruby][Rails]          line: 23 of 40, col: 1 (Top)
```

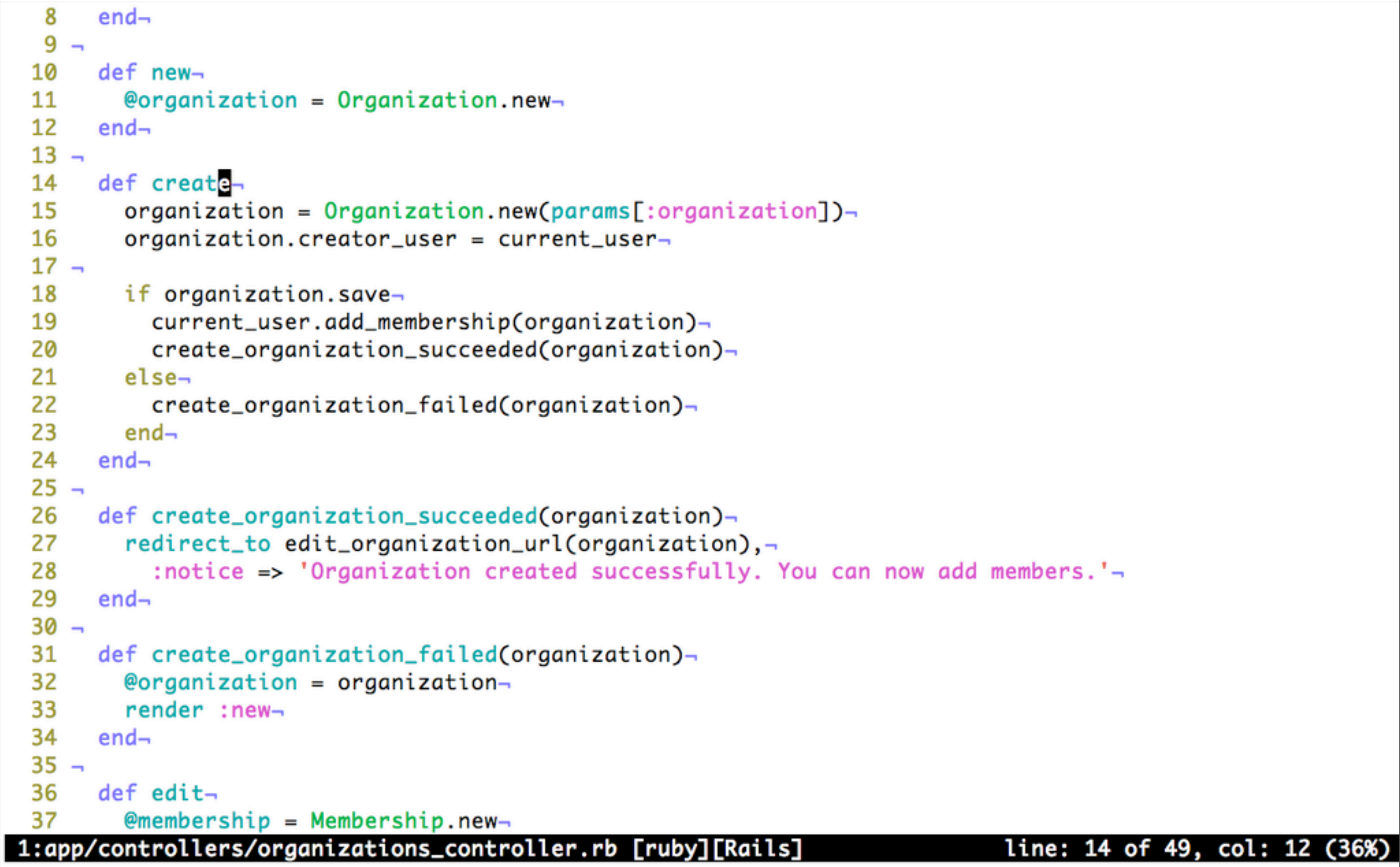


```
  end¬
¬
  def new¬
    @organization = Organization.new¬
  end¬
¬
  def create¬
    @organization = Organization.new(params[:organization])¬
    @organization.creator_user = current_user¬
¬
    if @organization.save¬
      current_user.add_membership(@organization)¬
      create_organization_succeeded¬
    else¬
      create_organization_failed¬
    end¬
  end¬
¬
  def create_organization_succeeded¬
    redirect_to edit_organization_url(@organization),¬
      :notice => 'Organization created successfully. You can now add members.'¬
  end¬
¬
  def create_organization_failed¬
    render :new¬
  end¬
¬
  def edit¬
    @membership = Membership.new¬
    @memberships = @organization.memberships.includes(:user)¬
 1:app/controllers/organizations_controller.rb [ruby][Rails]          line: 26 of 48, col: 1 (38%)
"app/controllers/organizations_controller.rb" 48L, 1114C written
```

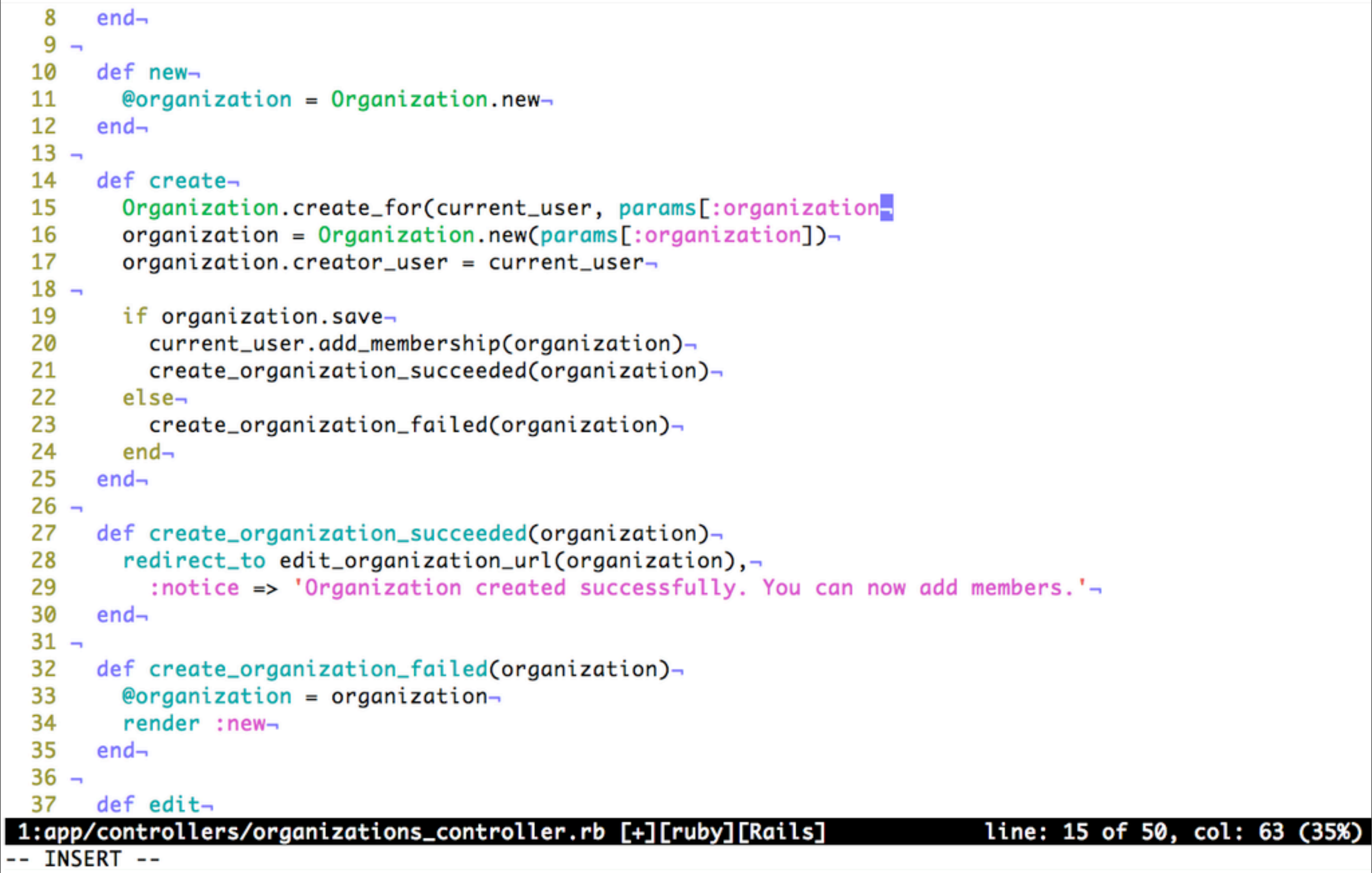


```ruby
  end

  def new
    @organization = Organization.new
  end

  def create
    @organization = Organization.new(params[:organization])
    @organization.creator_user = current_user

    if @organization.save
      current_user.add_membership(@organization)
      create_organization_succeeded
    else
      create_organization_failed
    end
  end

  def create_organization_succeeded
    redirect_to edit_organization_url(@organization),
      :notice => 'Organization created successfully. You can now add members.'
  end

  def create_organization_failed
    render :new
  end

  def edit
    @membership = Membership.new
    @memberships = @organization.memberships.includes(:user)
```

```
1:app/controllers/organizations_controller.rb [ruby][Rails]          line: 26 of 48, col: 1 (38%)
```

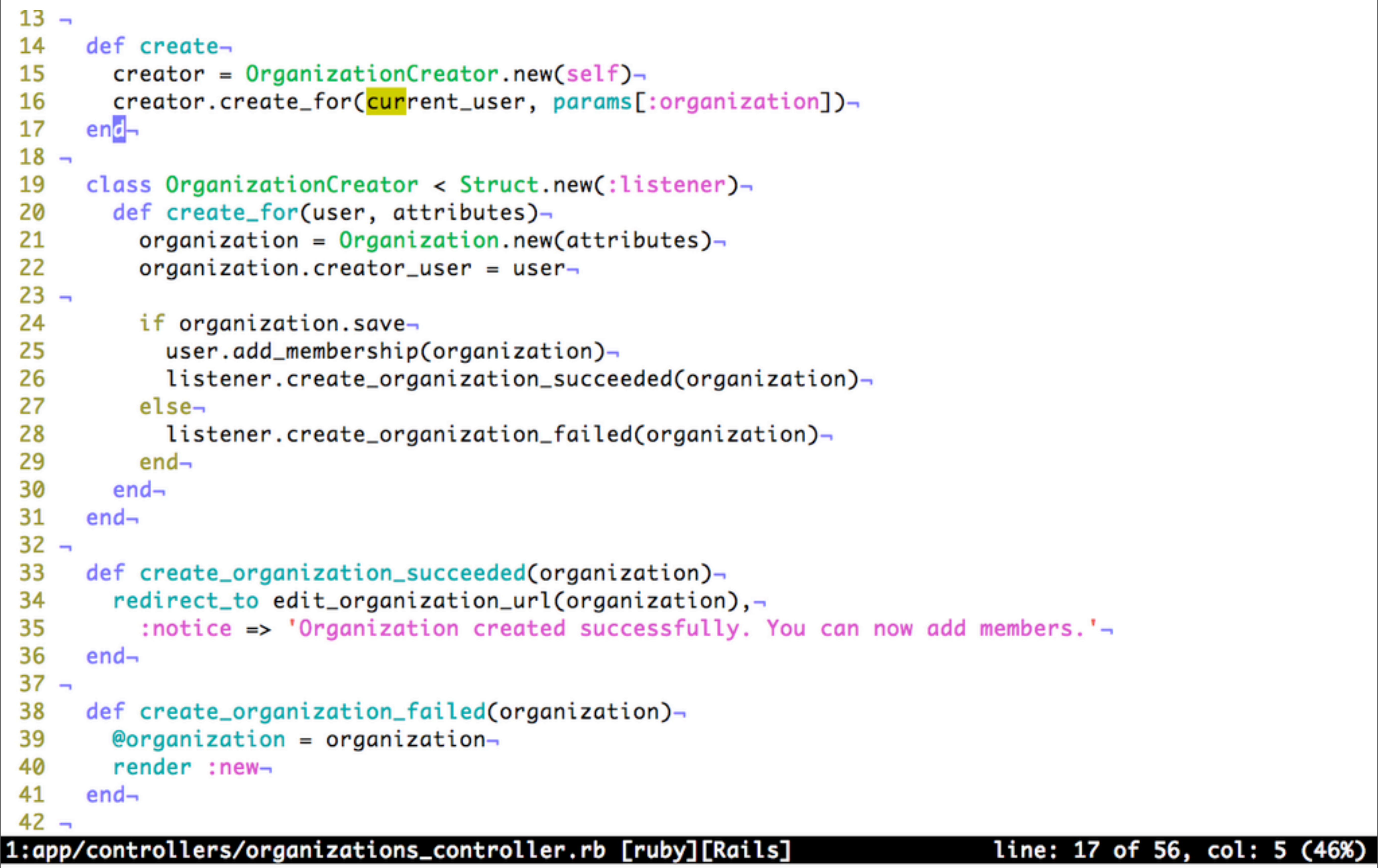


```
  end¬
¬
  def new¬
    @organization = Organization.new¬
  end¬
¬
  def create¬
    organization = Organization.new(params[:organization])¬
    organization.creator_user = current_user¬
¬
    if organization.save¬
      current_user.add_membership(organization)¬
      create_organization_succeeded(organization)¬
    else¬
      create_organization_failed(organization)¬
    end¬
  end¬
¬
  def create_organization_succeeded(organization)¬
    redirect_to edit_organization_url(organization),¬
      :notice => 'Organization created successfully. You can now add members.'¬
  end¬
¬
  def create_organization_failed(organization)¬
    @organization = organization¬
    render :new¬
  end¬
¬
  def edit¬
    @membership = Membership.new¬
1:app/controllers/organizations_controller.rb [ruby][Rails]          line: 14 of 49, col: 12 (36%)
```



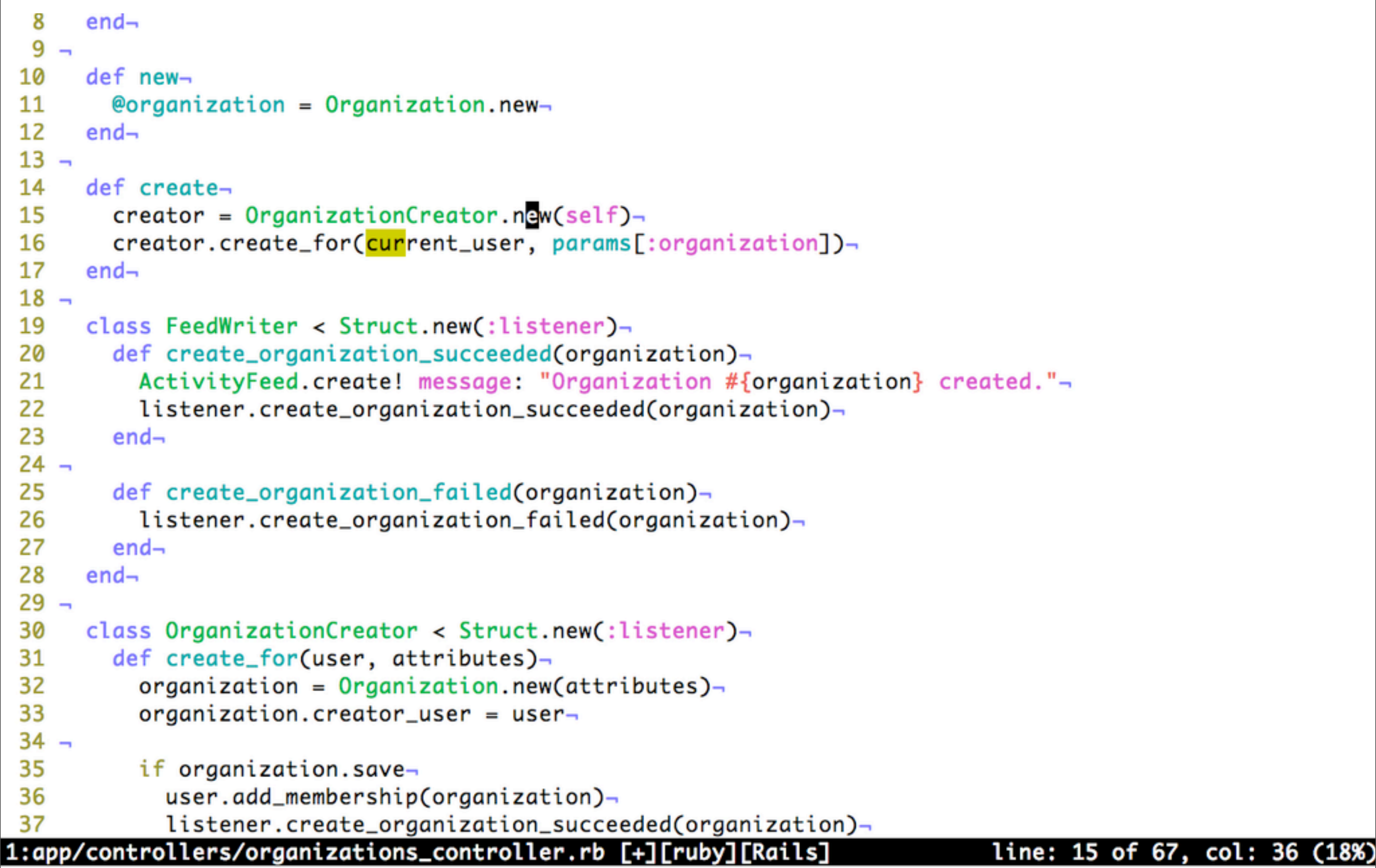
```ruby
  end

  def new
    @organization = Organization.new
  end

  def create
    Organization.create_for(current_user, params[:organization
    organization = Organization.new(params[:organization])
    organization.creator_user = current_user

    if organization.save
      current_user.add_membership(organization)
      create_organization_succeeded(organization)
    else
      create_organization_failed(organization)
    end
  end

  def create_organization_succeeded(organization)
    redirect_to edit_organization_url(organization),
      :notice => 'Organization created successfully. You can now add members.'
  end

  def create_organization_failed(organization)
    @organization = organization
    render :new
  end

  def edit
```

```
1:app/controllers/organizations_controller.rb [+][ruby][Rails]          line: 15 of 50, col: 63 (35%)
-- INSERT --
```



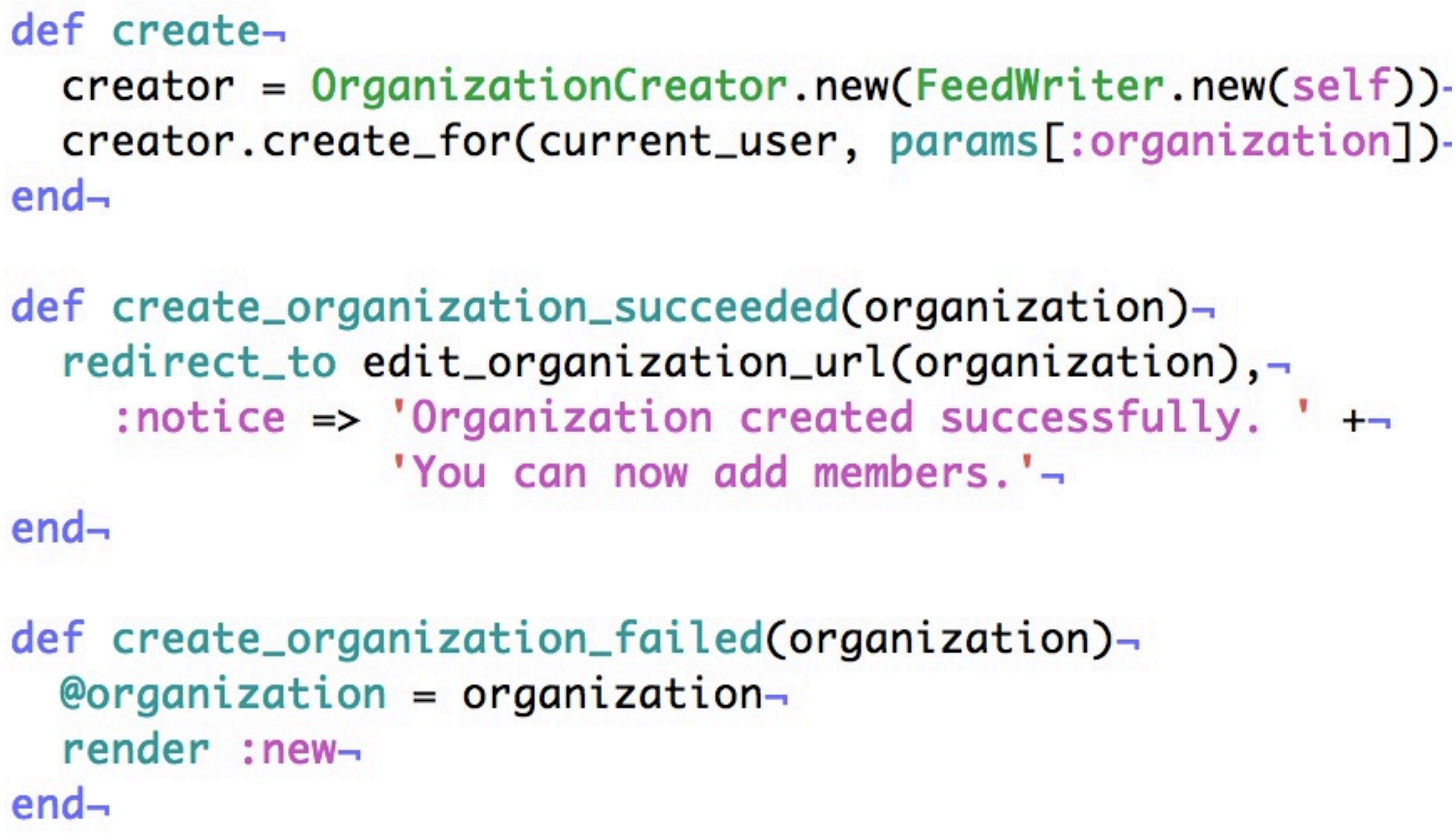
```ruby

  def create
    creator = OrganizationCreator.new(self)
    creator.create_for(current_user, params[:organization])
  end

  class OrganizationCreator < Struct.new(:listener)
    def create_for(user, attributes)
      organization = Organization.new(attributes)
      organization.creator_user = user

      if organization.save
        user.add_membership(organization)
        listener.create_organization_succeeded(organization)
      else
        listener.create_organization_failed(organization)
      end
    end
  end

  def create_organization_succeeded(organization)
    redirect_to edit_organization_url(organization),
      :notice => 'Organization created successfully. You can now add members.'
  end

  def create_organization_failed(organization)
    @organization = organization
    render :new
  end

```

```
1:app/controllers/organizations_controller.rb [ruby][Rails]          line: 17 of 56, col: 5 (46%)
```



```ruby
  end

  def new
    @organization = Organization.new
  end

  def create
    creator = OrganizationCreator.new(self)
    creator.create_for(current_user, params[:organization])
  end

  class FeedWriter < Struct.new(:listener)
    def create_organization_succeeded(organization)
      ActivityFeed.create! message: "Organization #{organization} created."
      listener.create_organization_succeeded(organization)
    end

    def create_organization_failed(organization)
      listener.create_organization_failed(organization)
    end
  end

  class OrganizationCreator < Struct.new(:listener)
    def create_for(user, attributes)
      organization = Organization.new(attributes)
      organization.creator_user = user

      if organization.save
        user.add_membership(organization)
        listener.create_organization_succeeded(organization)
```

```
1:app/controllers/organizations_controller.rb [+][ruby][Rails]          line: 15 of 67, col: 36 (18%)
```



```ruby
def create
  creator = OrganizationCreator.new(FeedWriter.new(self))
  creator.create_for(current_user, params[:organization])
end

def create_organization_succeeded(organization)
  redirect_to edit_organization_url(organization),
    :notice => 'Organization created successfully. ' +
               'You can now add members.'
end

def create_organization_failed(organization)
  @organization = organization
  render :new
end
```

Let's explore!

References

http://nccastaff.bournemouth.ac.uk/jmacey/CA1/Papers/Responsibility-Driven%20Design.pdf

http://www.wirfs-brock.com/PDFs/How%20Designs%20Differ.pdf

http://www.threeriversinstitute.org/blog/?p=338

http://alistair.cockburn.us/Hexagonal+architecture

http://www.growing-object-oriented-software.com/

http://pragprog.com/articles/tell-dont-ask

http://devblog.avdi.org/2012/03/15/now-available-objects-on-rails/

http://mattwynne.net/category/hexagonal-rails/