
an Schopohl Photography


Steglitz
Grazer Damm
Tiergarten
Sechserdamm


POST


Dreieck
Hannover-Nord
7
7

Big or Small?

Stand up if in your codebase you…

- have more than 50 models
- have a class with more than a 1000 LOC
- have an ActiveRecord model with more than 30 :has_many
- can’t run your full test suite on your machine
- can’t run your test suite on your machine in under 20 minutes

Structural Engineering in Ruby

Stephan Hagemann

The *only* thing that ultimately
let's you reign in complexity
within one app are *components*

As a language, Ruby
does not have
components

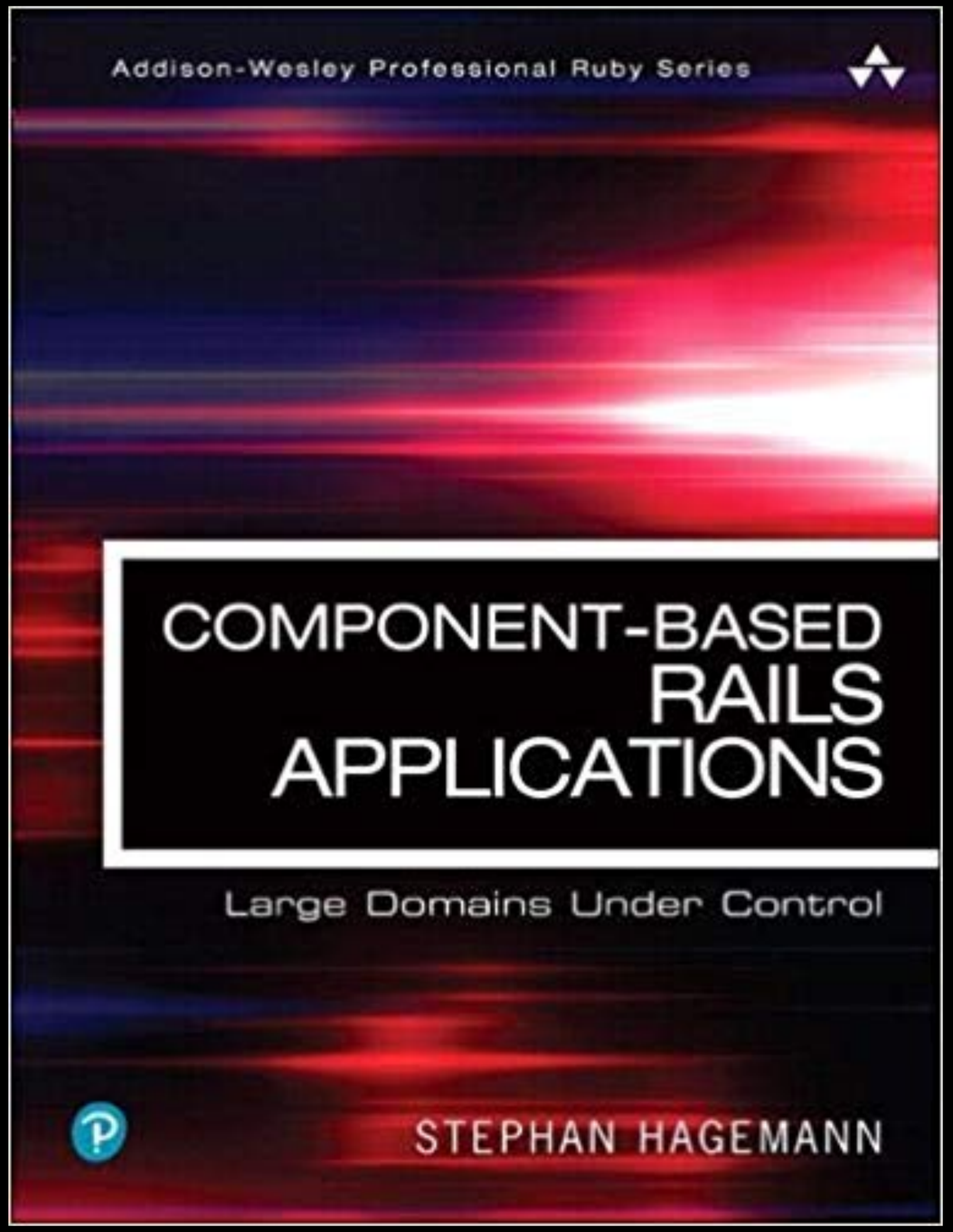
Gems can act as components

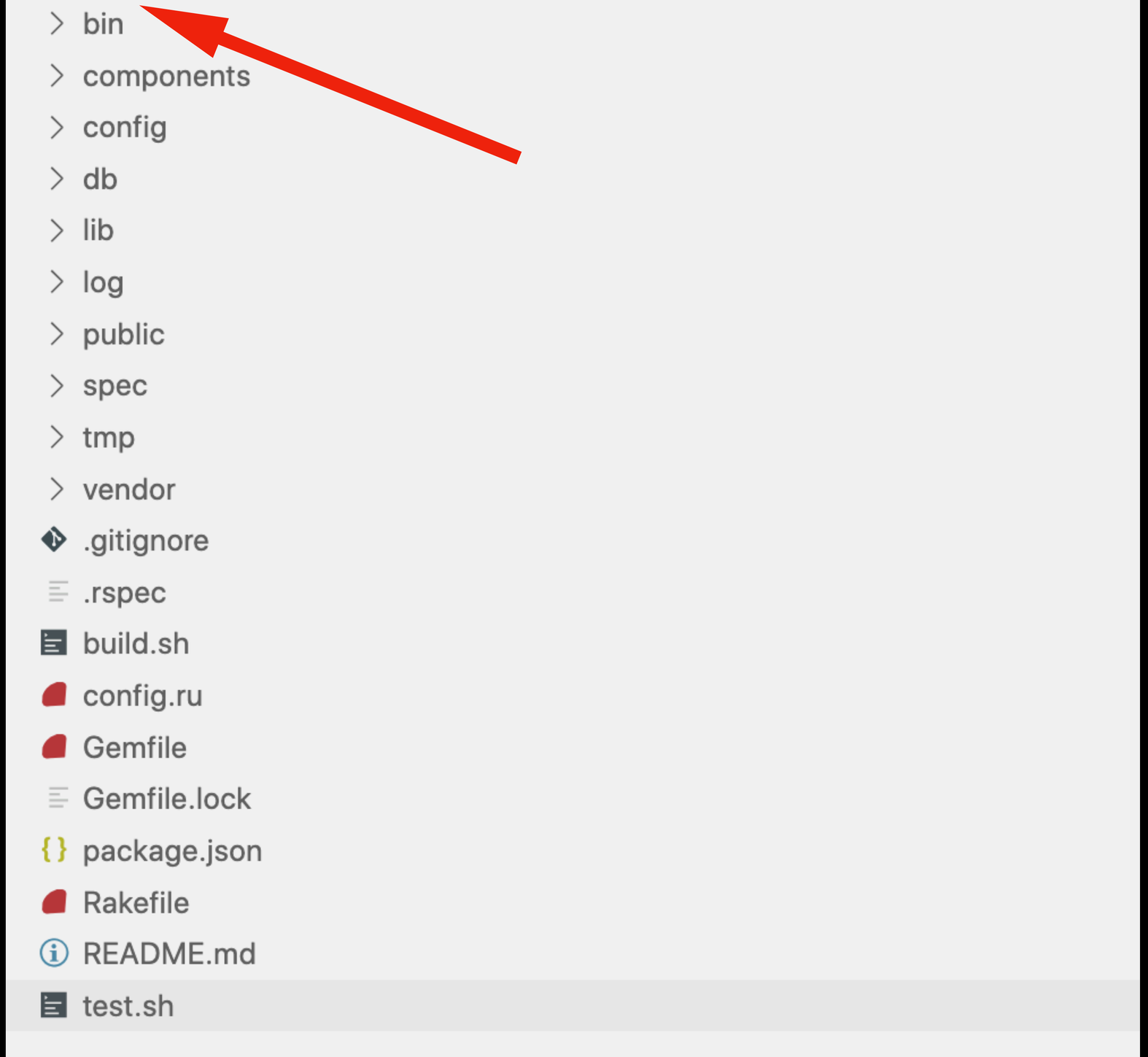
…they work well

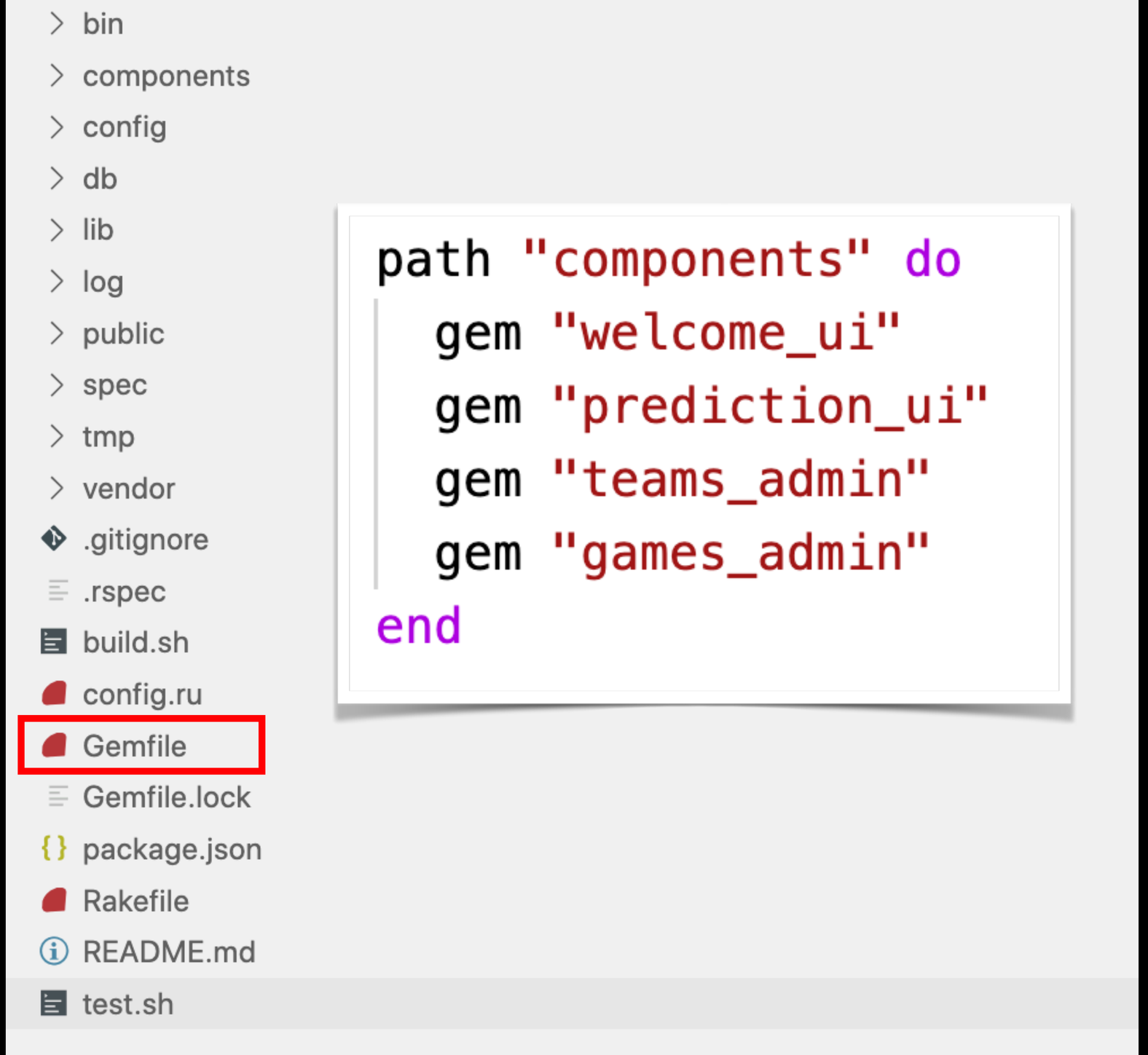
For Rails, the situation kinda sucks…

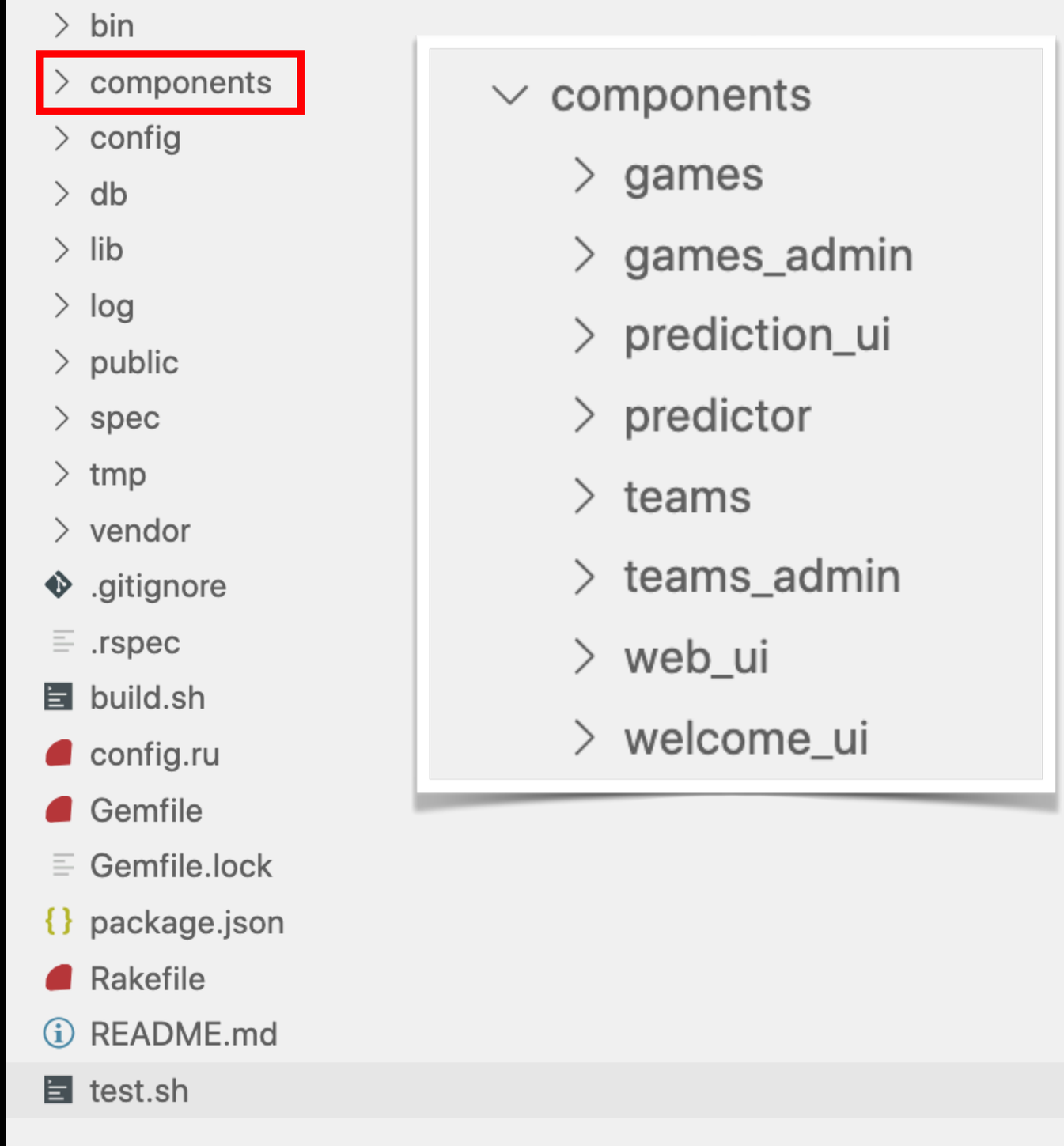
…but works ok

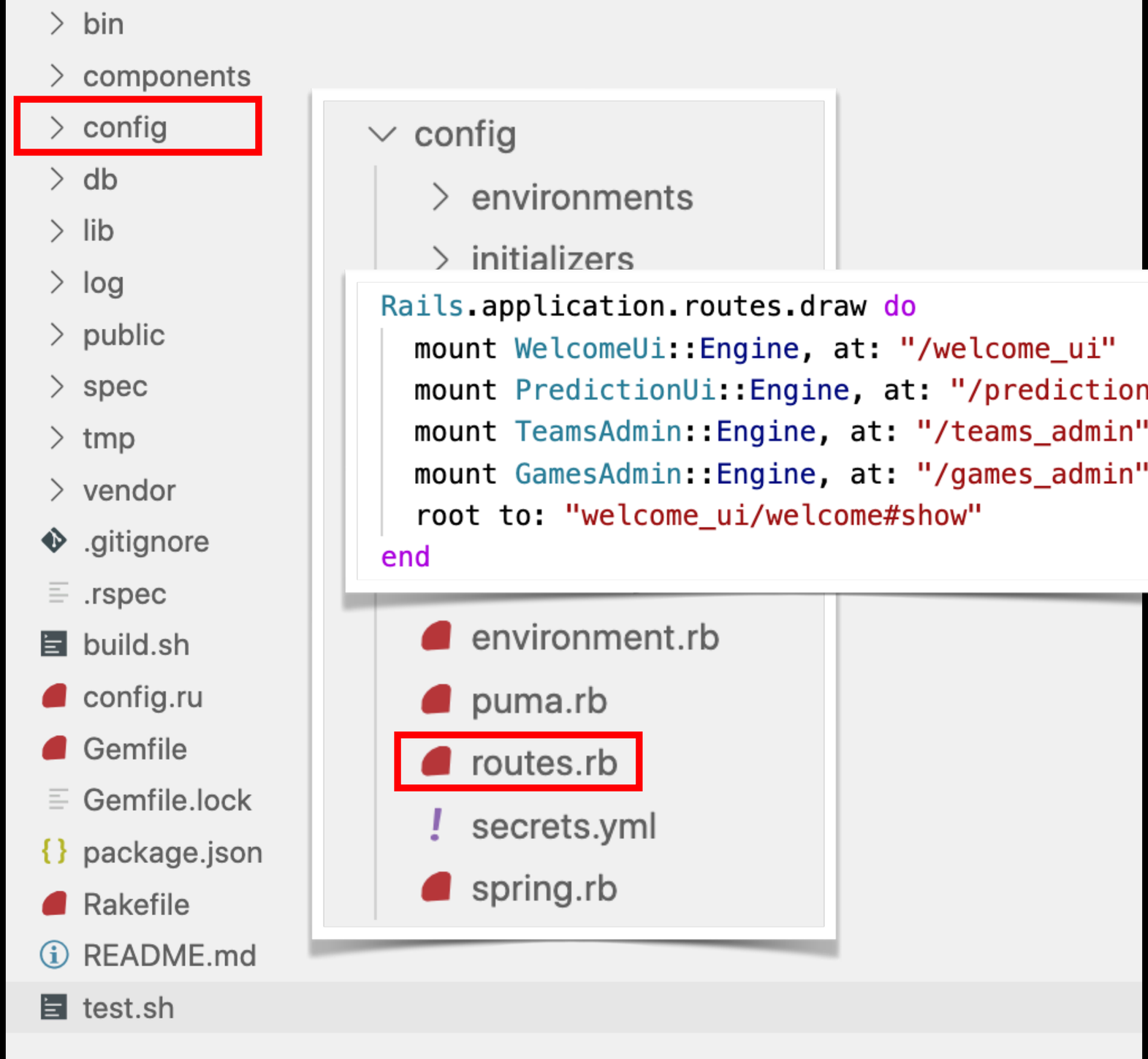
Let’s make it better!

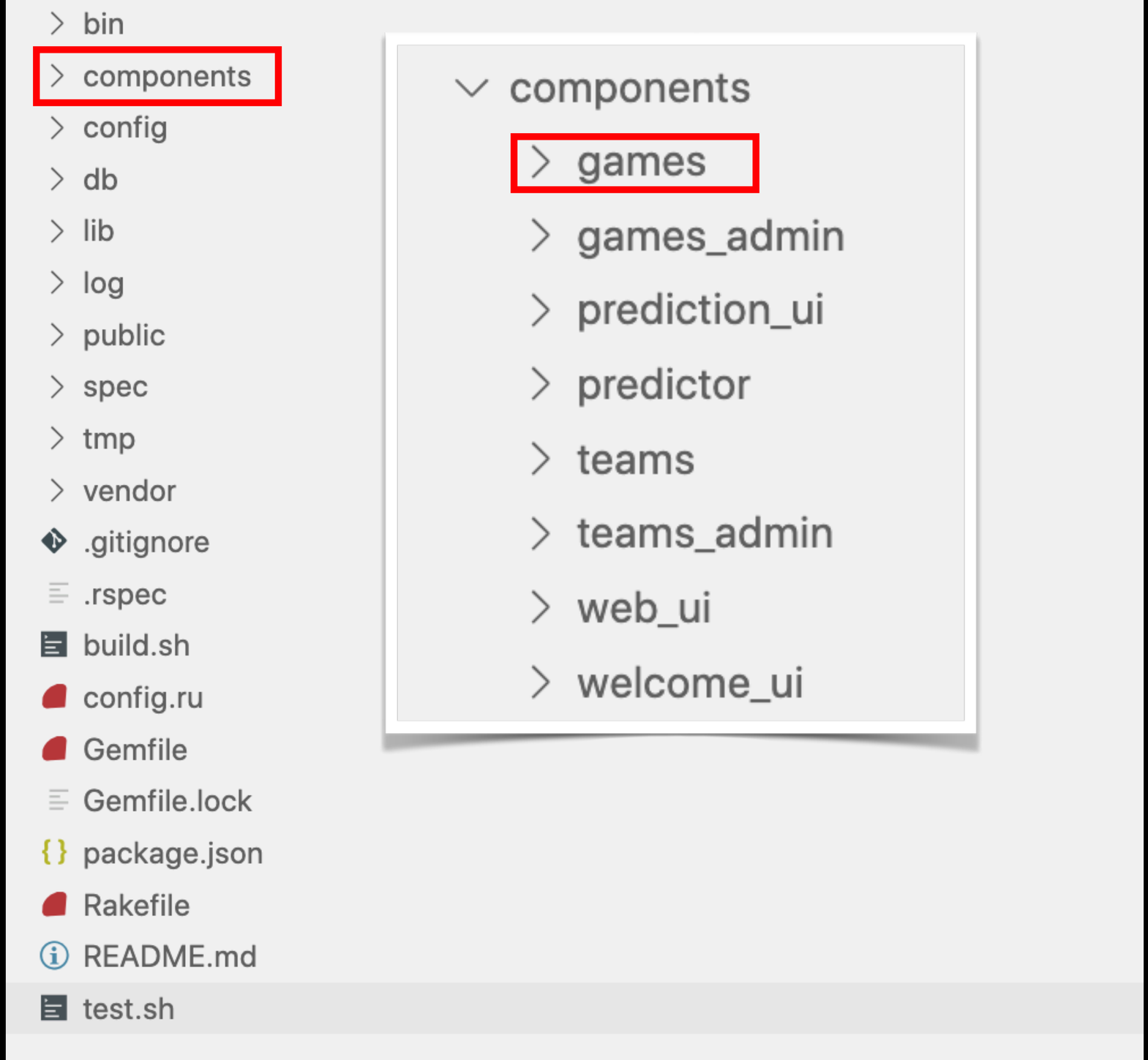

Addison-Wesley Professional Ruby Series
COMPONENT-BASED RAILS APPLICATIONS
Large Domains Under Control
STEPHAN HAGEMANN

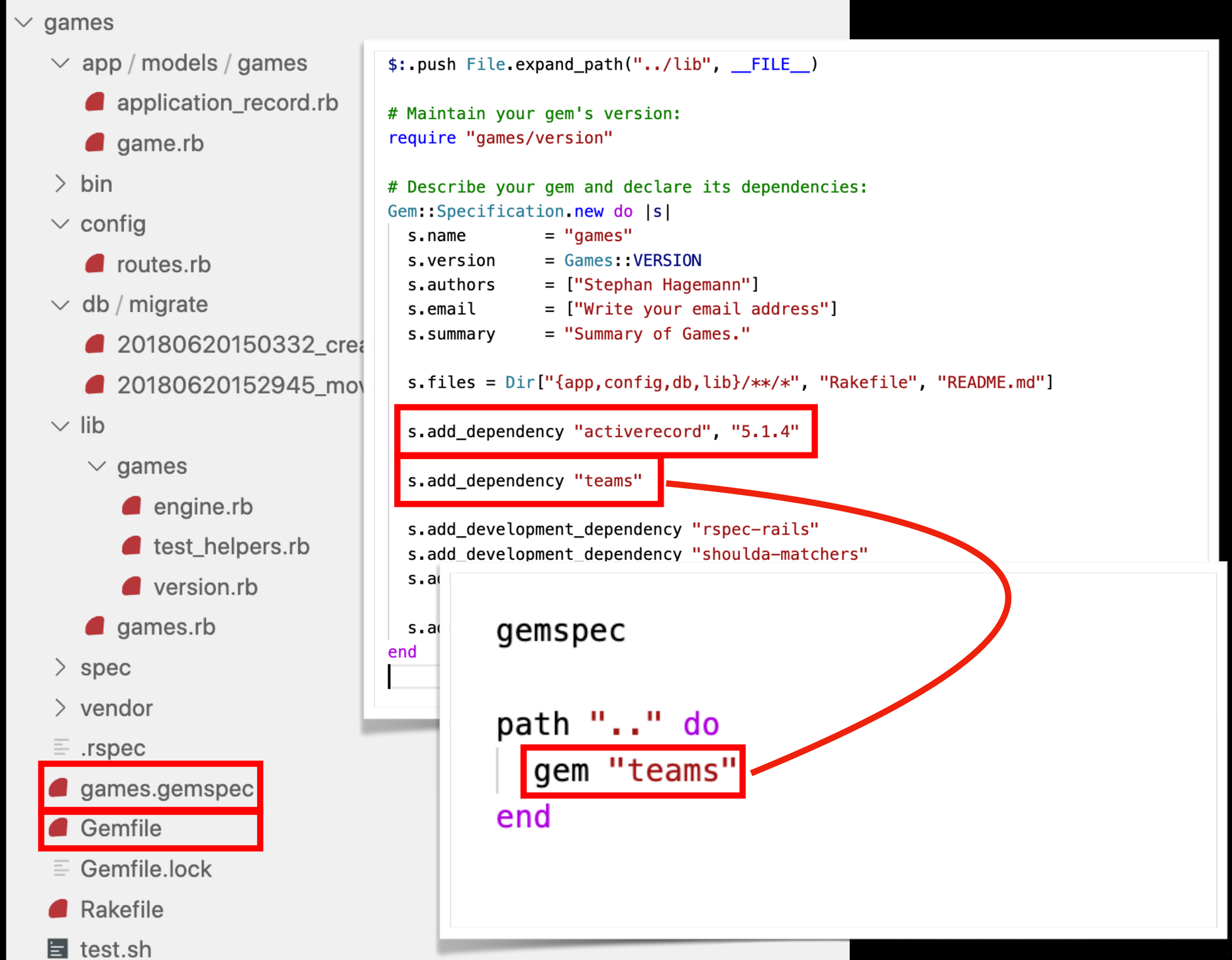

bin
components
config
db
lib
log
public
spec
tmp
vendor
.gitignore
.rspec
build.sh
config.ru
Gemfile
Gemfile.lock
package.json
Rakefile
README.md
test.sh


bin
components
config
db
lib
log
public
spec
tmp
vendor
.gitignore
.rspec
build.sh
config.ru
Gemfile
Gemfile.lock
package.json
Rakefile
README.md
test.sh
path "components" do
gem "welcome_ui"
gem "prediction_ui"
gem "teams_admin"
gem "games_admin"
end

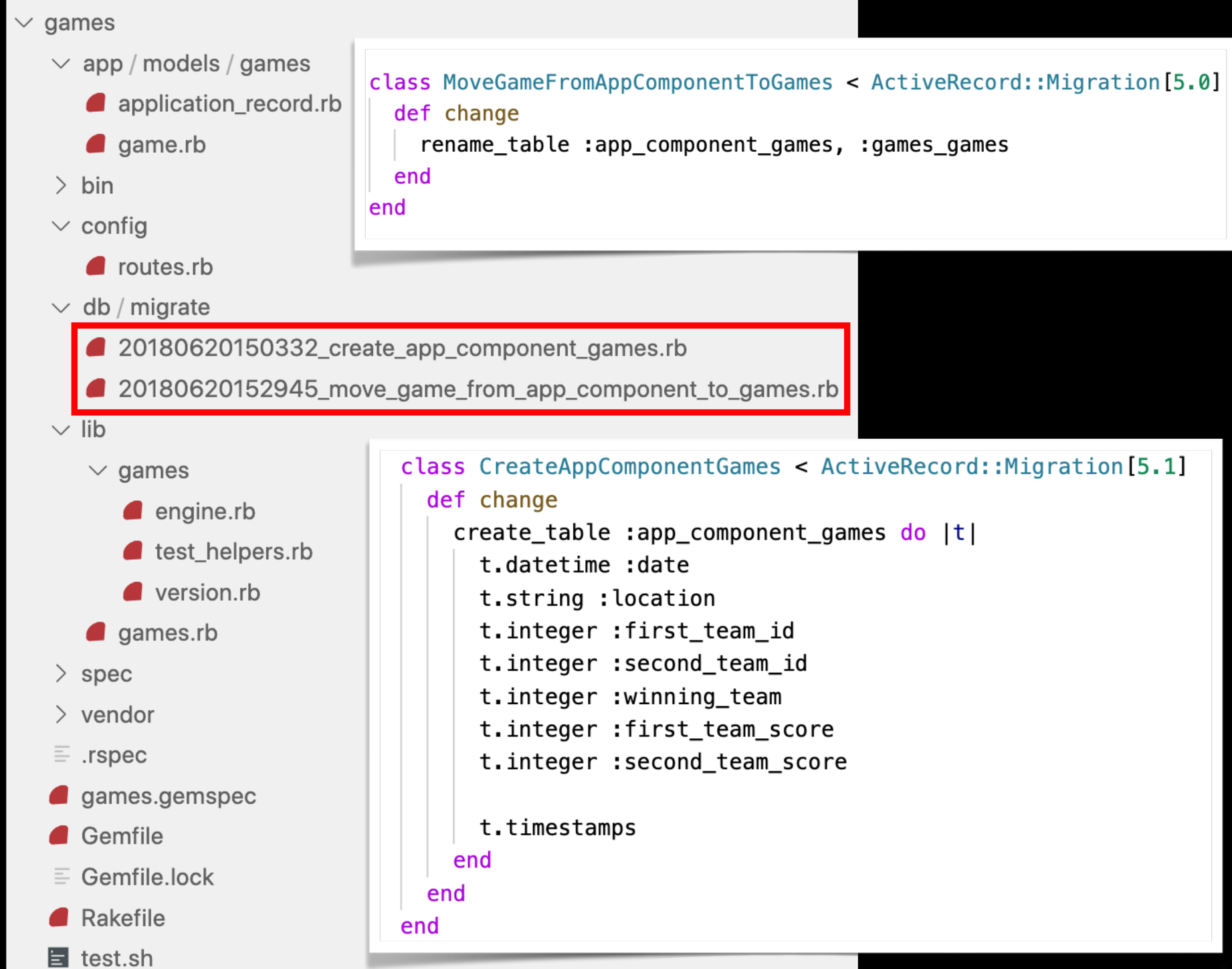

bin
components
config
db
lib
log
public
spec
tmp
vendor
.gitignore
.rspec
build.sh
config.ru
Gemfile
Gemfile.lock
package.json
Rakefile
README.md
test.sh
components
games
games_admin
prediction_ui
predictor
teams
teams_admin
web_ui
welcome_ui

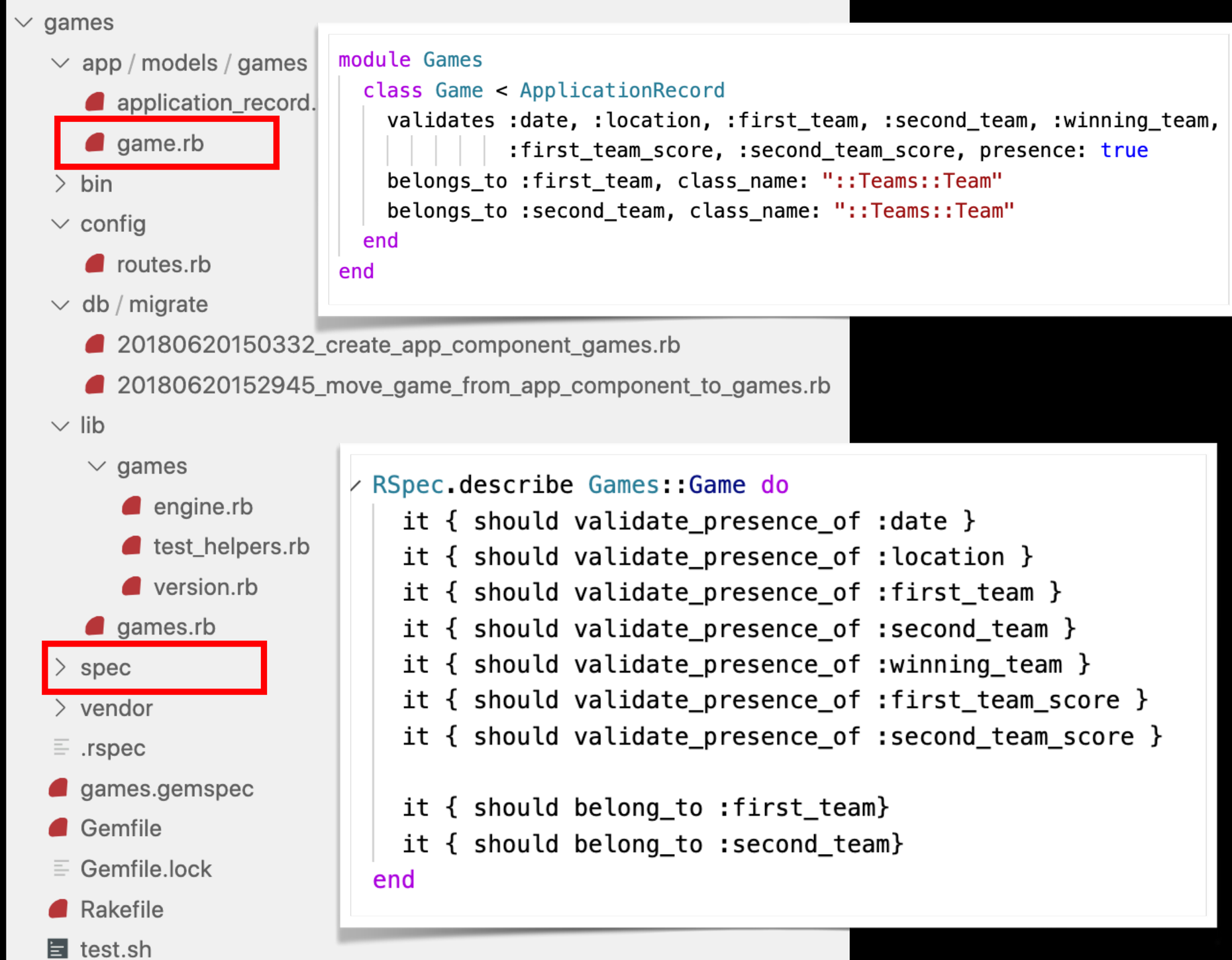

bin
components
config
db
lib
log
public
spec
tmp
vendor
.gitignore
.rspec
build.sh
config.ru
Gemfile
Gemfile.lock
package.json
Rakefile
README.md
test.sh
config
environments
initializers
Rails.application.routes.draw do
mount WelcomeUi::Engine, at: "/welcome_ui"
mount PredictionUi::Engine, at: "/prediction_ui"
mount TeamsAdmin::Engine, at: "/teams_admin"
mount GamesAdmin::Engine, at: "/games_admin"
root to: "welcome_ui/welcome#show"
end
environment.rb
puma.rb
routes.rb
secrets.yml
spring.rb

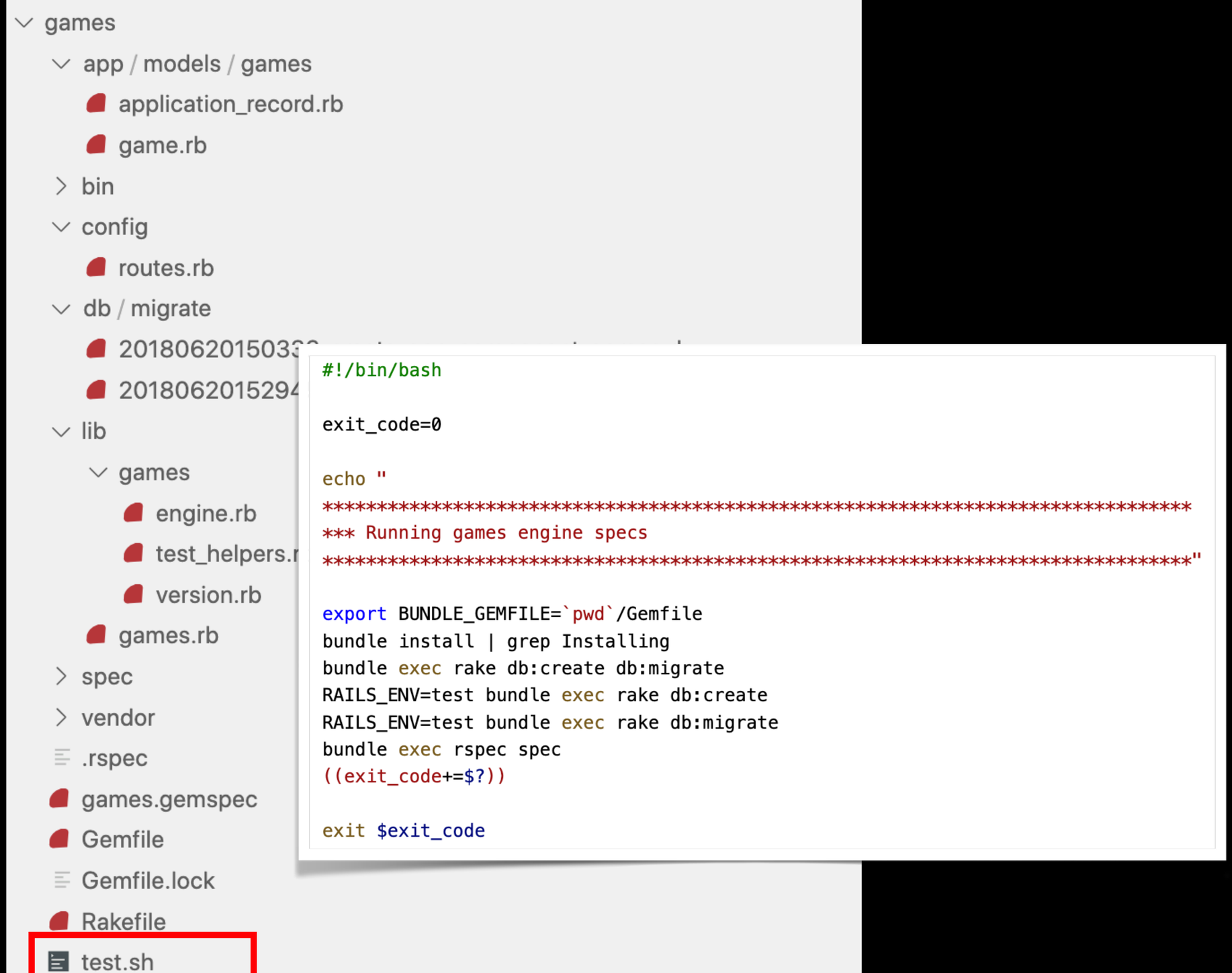

bin
components
config
db
lib
log
public
spec
tmp
vendor
.gitignore
.rspec
build.sh
config.ru
Gemfile
Gemfile.lock
package.json
Rakefile
README.md
test.sh
components
games
games_admin
prediction_ui
predictor
teams
teams_admin
web_ui
welcome_ui

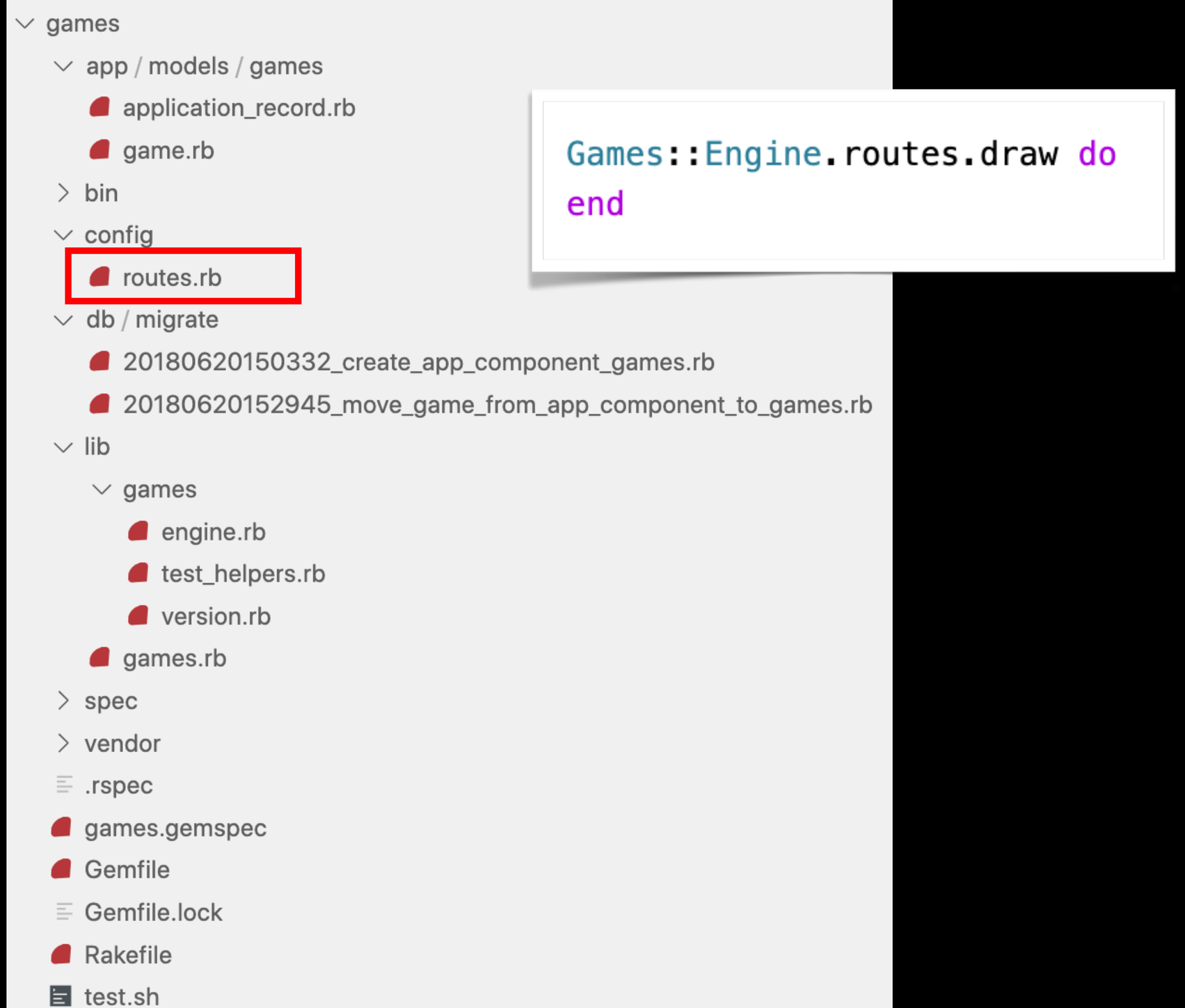

games
app / models / games
application_record.rb
game.rb
bin
config
routes.rb
db / migrate
20180620150332_crea
20180620152945_mov
lib
games
engine.rb
test_helpers.rb
version.rb
games.rb
spec
vendor
.rspec
games.gemspec
Gemfile
Gemfile.lock
Rakefile
test.sh
$:.push File.expand_path("../lib", __FILE__)
Maintain your gem's version:
require "games/version"
Describe your gem and declare its dependencies:
Gem::Specification.new do |s|
s.name = "games"
s.version = Games::VERSION
s.authors = ["Stephan Hagemann"]
s.email = ["Write your email address"]
s.summary = "Summary of Games."
s.files = Dir["{app,config,db,lib}/**/*", "Rakefile", "README.md"]
s.add_dependency "activerecord", "5.1.4"
s.add_dependency "teams"
s.add_development_dependency "rspec-rails"
s.add_development_dependency "shoulda-matchers"
end
gemspec
path ".." do
gem "teams"
end

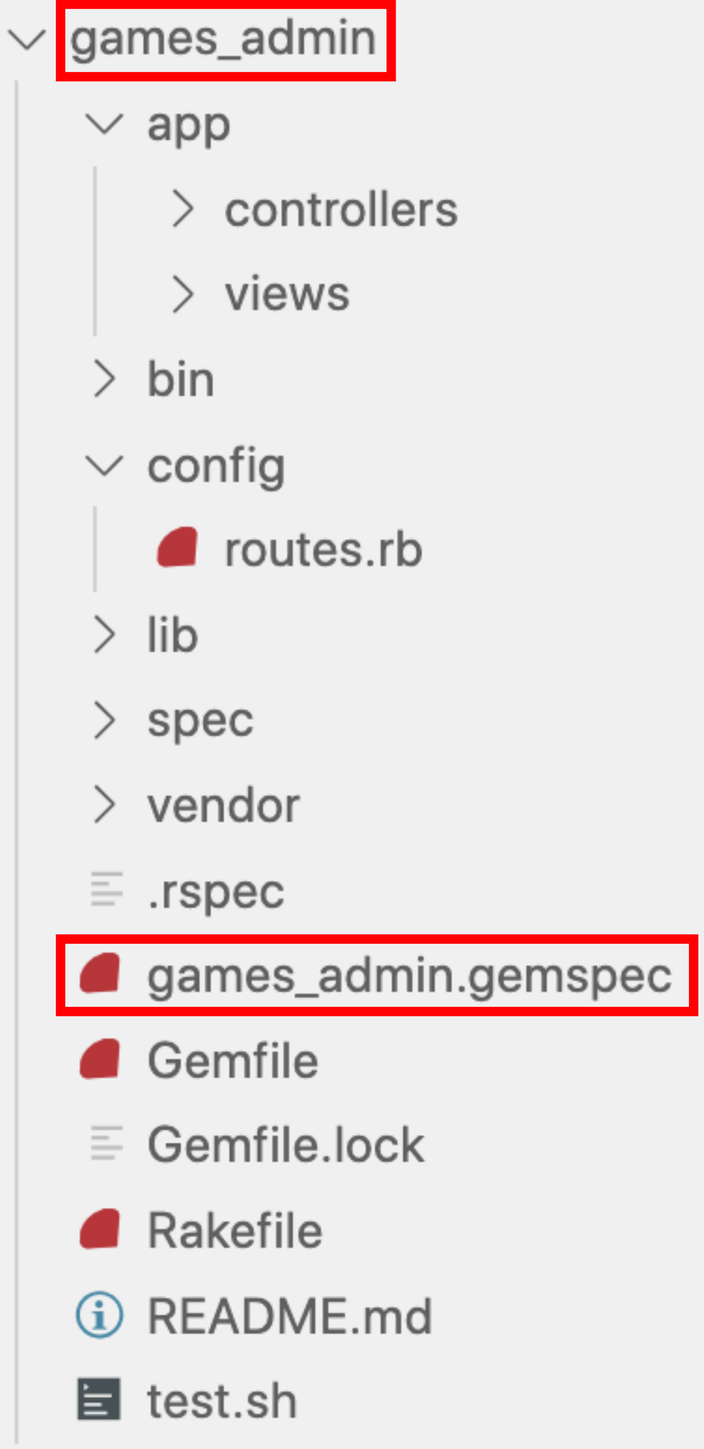

games
app / models / games
application_record.rb
game.rb
bin
config
routes.rb
db / migrate
20180620150332_create_app_component_games.rb
20180620152945_move_game
lib
games
engine.rb
test_helpers.rb
version.rb
games.rb
spec
vendor
.rspec
games.gemspec
Gemfile
Gemfile.lock
Rakefile
test.sh
require "games/engine"
module Games
end
module Games
class Engine < ::Rails::Engine
isolate_namespace Games
initializer :append_migrations do |app|
unless app.root.to_s.match root.to_s+File::SEPARATOR
app.config.paths["db/migrate"].concat(
config.paths["db/migrate"].expanded
)
end
end
config.generators do |g|
g.orm :active_record
g.test_framework :rspec
end
end
end


games
app / models / games
application_record.rb
game.rb
bin
config
routes.rb
db / migrate
20180620150332_create_app_component_games.rb
20180620152945_move_game_from_app_component_to_games.rb
lib
games
engine.rb
test_helpers.rb
version.rb
games.rb
spec
vendor
.rspec
games.gemspec
Gemfile
Gemfile.lock
Rakefile
test.sh
class MoveGameFromAppComponentToGames < ActiveRecord::Migration[5.0]
def change
rename_table :app_component_games, :games_games
end
end
class CreateAppComponentGames < ActiveRecord::Migration[5.1]
def change
create_table :app_component_games do |t|
t.datetime :date
t.string :location
t.integer :first_team_id
t.integer :second_team_id
t.integer :winning_team
t.integer :first_team_score
t.integer :second_team_score
t.timestamps
end
end
end

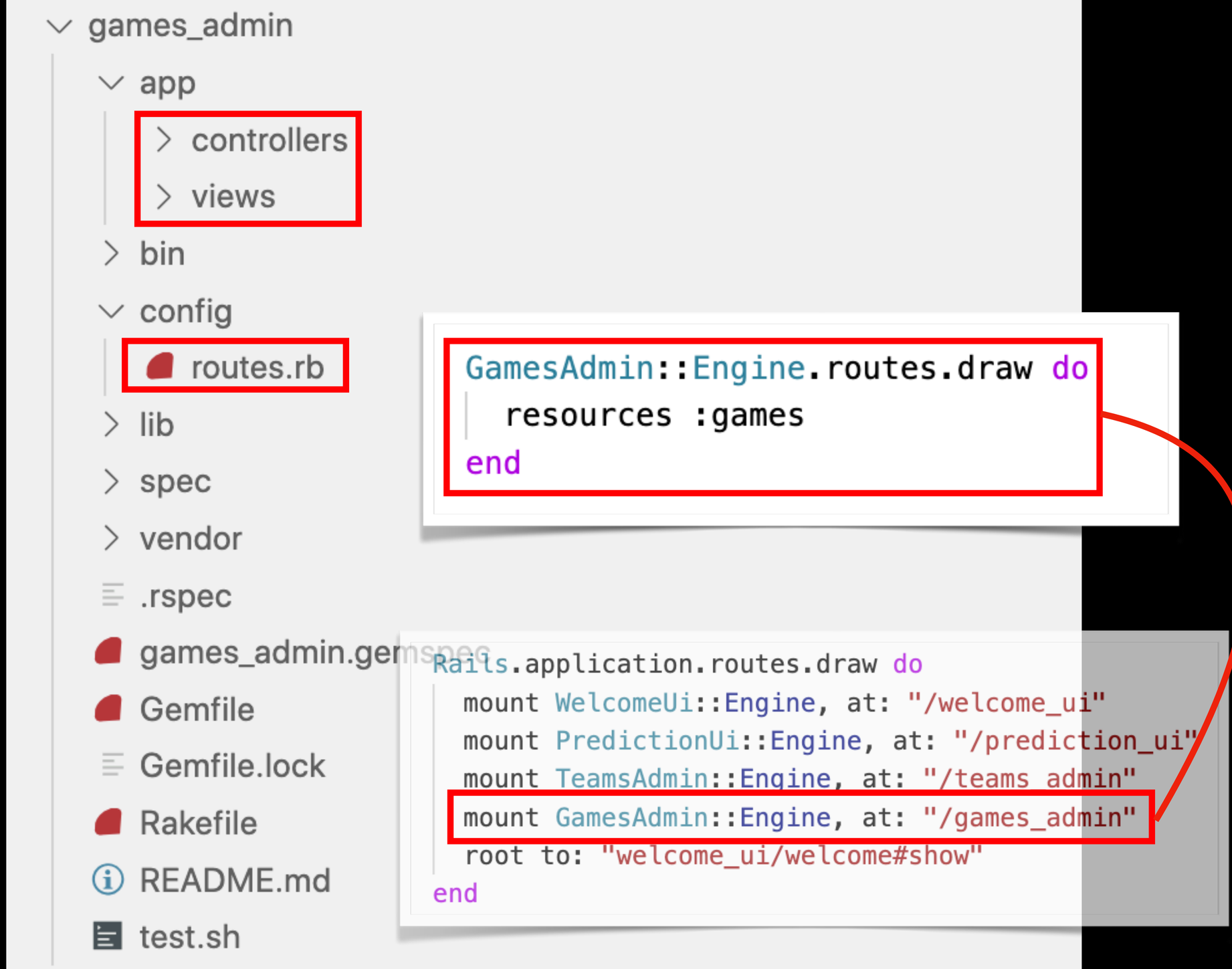

games
app / models / games
application_record.
game.rb
bin
config
routes.rb
db / migrate
20180620150332_create_app_component_games.rb
20180620152945_move_game_from_app_component_to_games.rb
lib
games
engine.rb
test_helpers.rb
version.rb
games.rb
spec
vendor
.rspec
games.gemspec
Gemfile
Gemfile.lock
Rakefile
test.sh
module Games
class Game < ApplicationRecord
validates :date, :location, :first_team, :second_team, :winning_team,
:first_team_score, :second_team_score, presence: true
belongs_to :first_team, class_name: "::Teams::Team"
belongs_to :second_team, class_name: "::Teams::Team"
end
end
RSpec.describe Games::Game do
it { should validate_presence_of :date }
it { should validate_presence_of :location }
it { should validate_presence_of :first_team }
it { should validate_presence_of :second_team }
it { should validate_presence_of :winning_team }
it { should validate_presence_of :first_team_score }
it { should validate_presence_of :second_team_score }
it { should belong_to :first_team}
it { should belong_to :second_team}
end

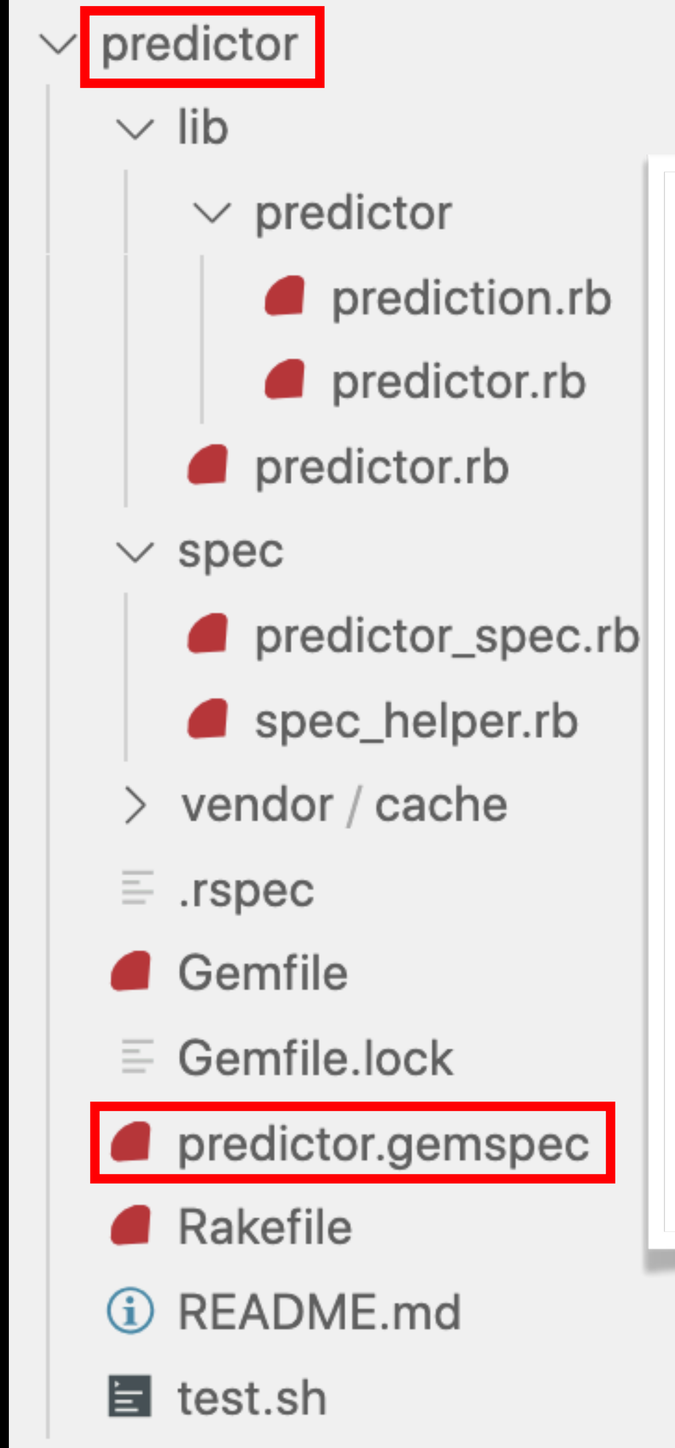

games
app / models / games
application_record.rb
game.rb
bin
config
routes.rb
db / migrate
lib
games
engine.rb
version.rb
games.rb
spec
vendor
.rspec
games.gemspec
Gemfile
Gemfile.lock
Rakefile
test.sh
#!/bin/bash
exit_code=0
echo "
*** Running games engine specs
"
export BUNDLE_GEMFILE=`pwd`/Gemfile
bundle install | grep Installing
bundle exec rake db:create db:migrate
RAILS_ENV=test bundle exec rake db:create
RAILS_ENV=test bundle exec rake db:migrate
bundle exec rspec spec
((exit_code+=$?))
exit $exit_code


games
app / models / games
application_record.rb
game.rb
bin
config
routes.rb
db / migrate
20180620150332_create_app_component_games.rb
20180620152945_move_game_from_app_component_to_games.rb
lib
games
engine.rb
test_helpers.rb
version.rb
games.rb
spec
vendor
.rspec
games.gemspec
Gemfile
Gemfile.lock
Rakefile
test.sh
Games::Engine.routes.draw do
end

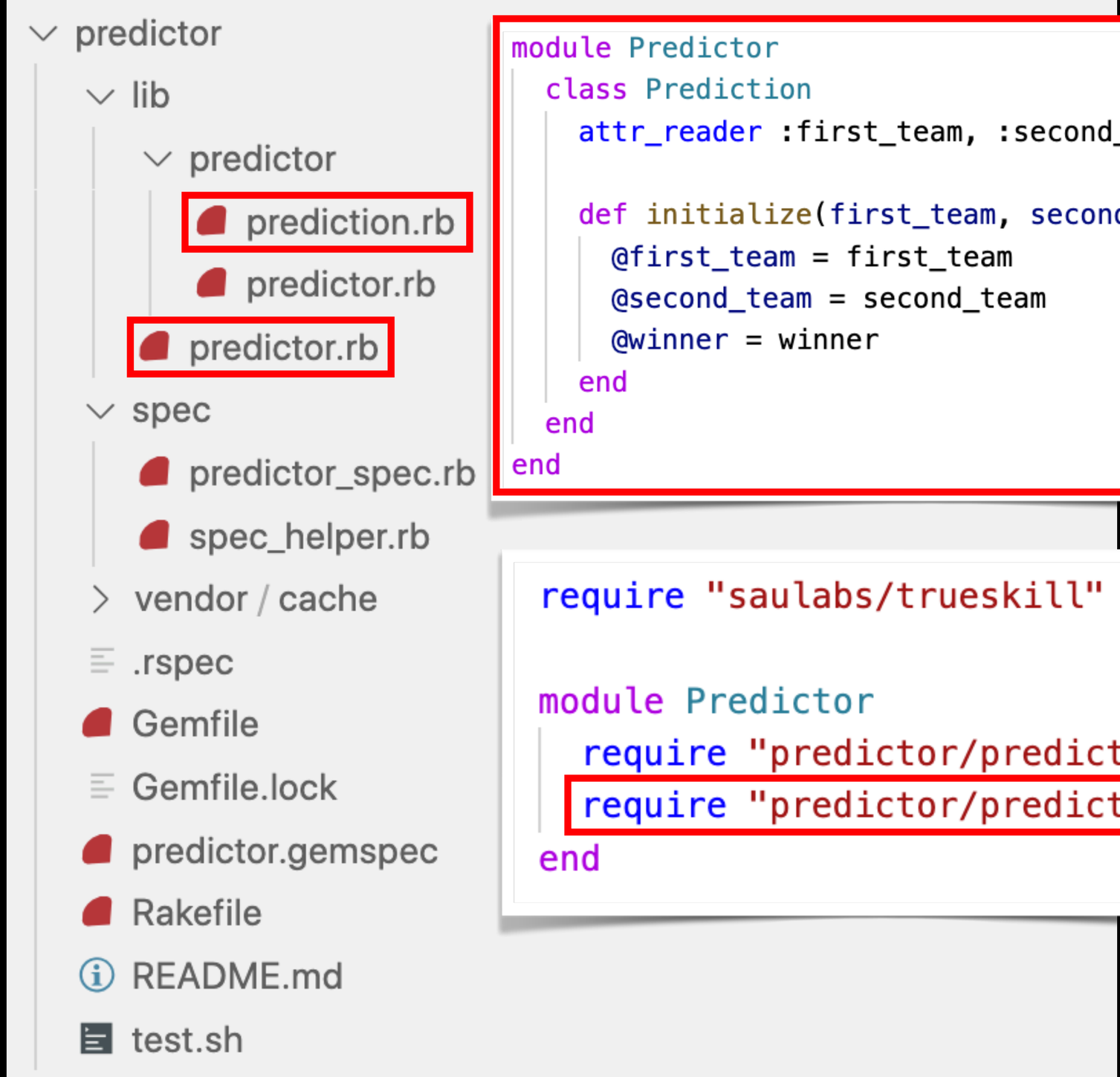



```ruby
$:.push File.expand_path("../lib", __FILE__)

# Maintain your gem's version:
require "games_admin/version"

# Describe your gem and declare its dependencies:
Gem::Specification.new do |s|
  s.name        = "games_admin"
  s.version     = GamesAdmin::VERSION
  s.authors     = ["Stephan Hagemann"]
  s.email       = ["Write your email address"]
  s.summary     = "Summary of GamesAdmin."

  s.files = Dir["{app,config,db,lib}/**/*", "Rakefile", "README.md"]

  s.add_dependency "rails", "5.1.4"
  s.add_dependency "games"
  s.add_dependency "teams"
  s.add_dependency "slim-rails", "3.1.3"
  s.add_dependency "jquery-rails", "4.3.1"

  s.add_dependency "web_ui"

  s.add_development_dependency "rspec-rails"
  s.add_development_dependency "shoulda-matchers"
  s.add_development_dependency "database_cleaner"
  s.add_development_dependency "capybara"
  s.add_development_dependency "rails-controller-testing"


  s.add_development_dependency "sqlite3"
end
```


games_admin
app
controllers
views
bin
config
routes.rb
lib
spec
vendor
.rspec
games_admin.gemspec
Gemfile
Gemfile.lock
Rakefile
README.md
test.sh
GamesAdmin::Engine.routes.draw do
resources :games
end
Rails.application.routes.draw do
mount WelcomeUi::Engine, at: "/welcome_ui"
mount PredictionUi::Engine, at: "/prediction_ui"
mount TeamsAdmin::Engine, at: "/teams_admin"
mount GamesAdmin::Engine, at: "/games_admin"
root to: "welcome_ui/welcome#show"
end

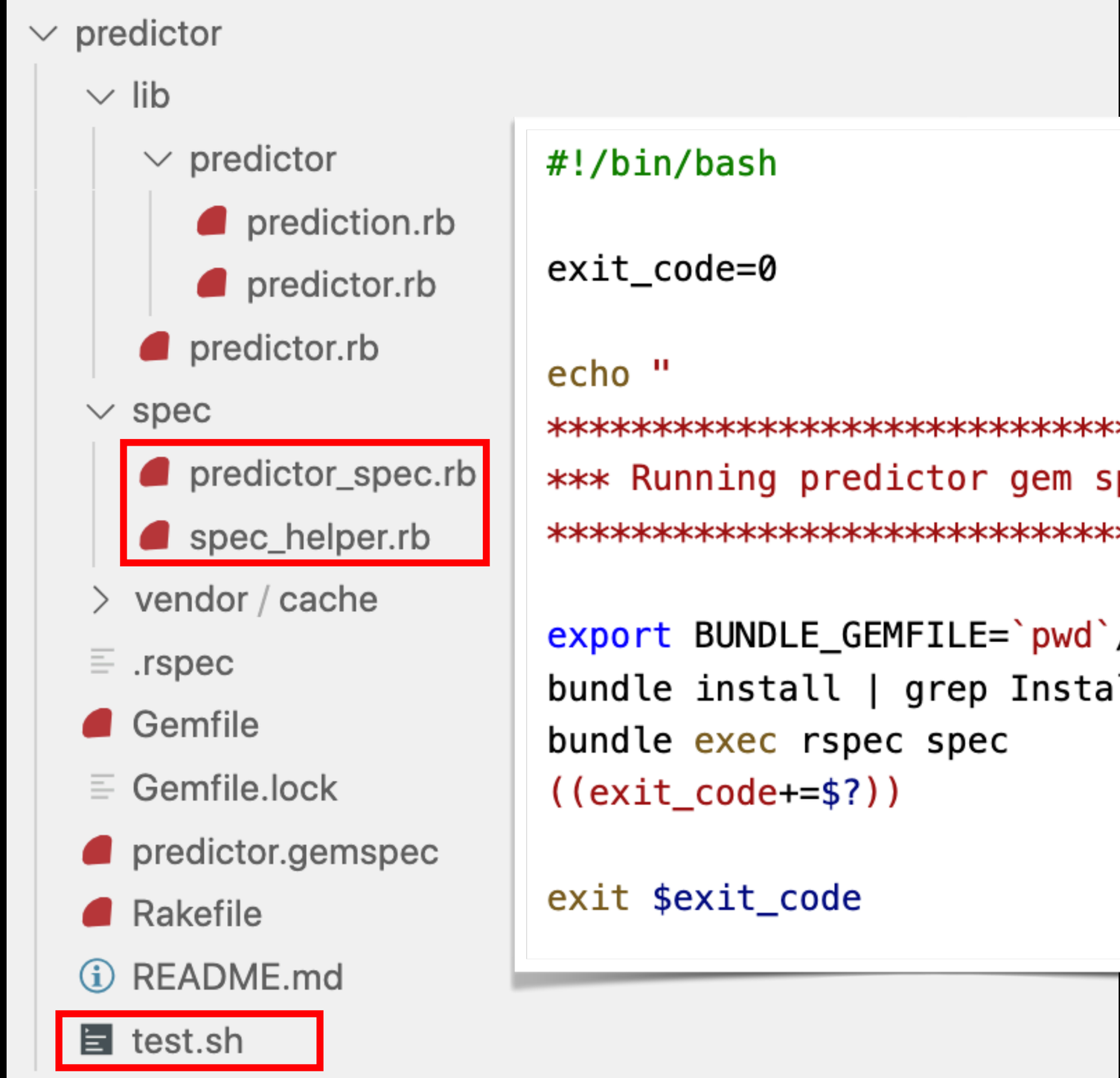



```ruby
# coding: utf-8
lib = File.expand_path("../lib", __FILE__)
$LOAD_PATH.unshift(lib) unless $LOAD_PATH.include?(lib)

Gem::Specification.new do |spec|
  spec.name          = "predictor"
  spec.version       = "0.1.0"
  spec.authors       = ["Stephan Hagemann"]
  spec.email         = ["stephan.hagemann@gmail.com"]

  spec.summary       = %q{Prediction Core}

  spec.files = Dir["{lib}/**/*", "Rakefile", "README.md"]
  spec.require_paths = ["lib"]

  spec.add_dependency "trueskill"

  spec.add_development_dependency "bundler"
  spec.add_development_dependency "rake"
  spec.add_development_dependency "rspec"
end
```

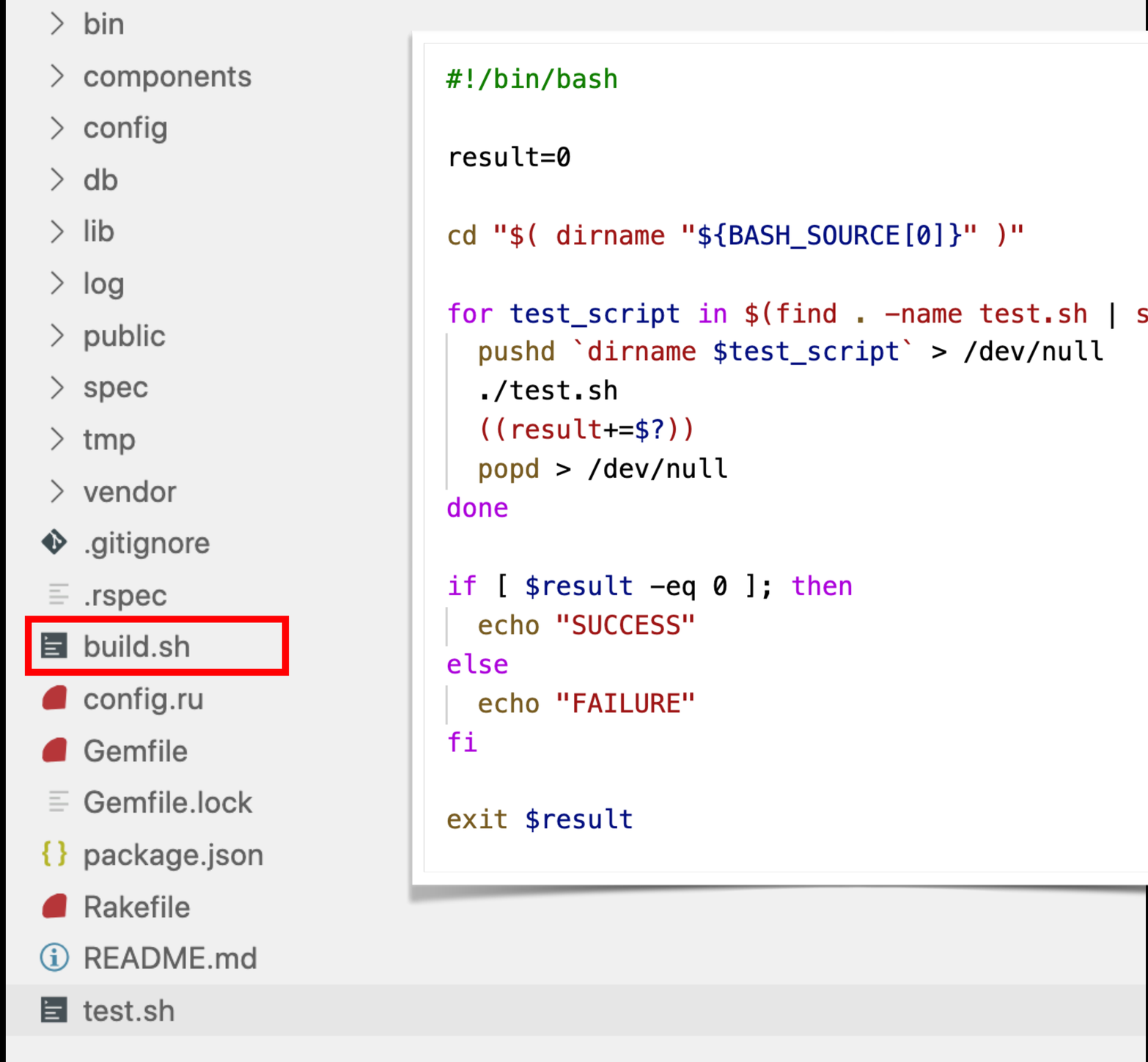



```ruby
module Predictor
  class Prediction
    attr_reader :first_team, :second_team, :winner

    def initialize(first_team, second_team, winner)
      @first_team = first_team
      @second_team = second_team
      @winner = winner
    end
  end
end
```

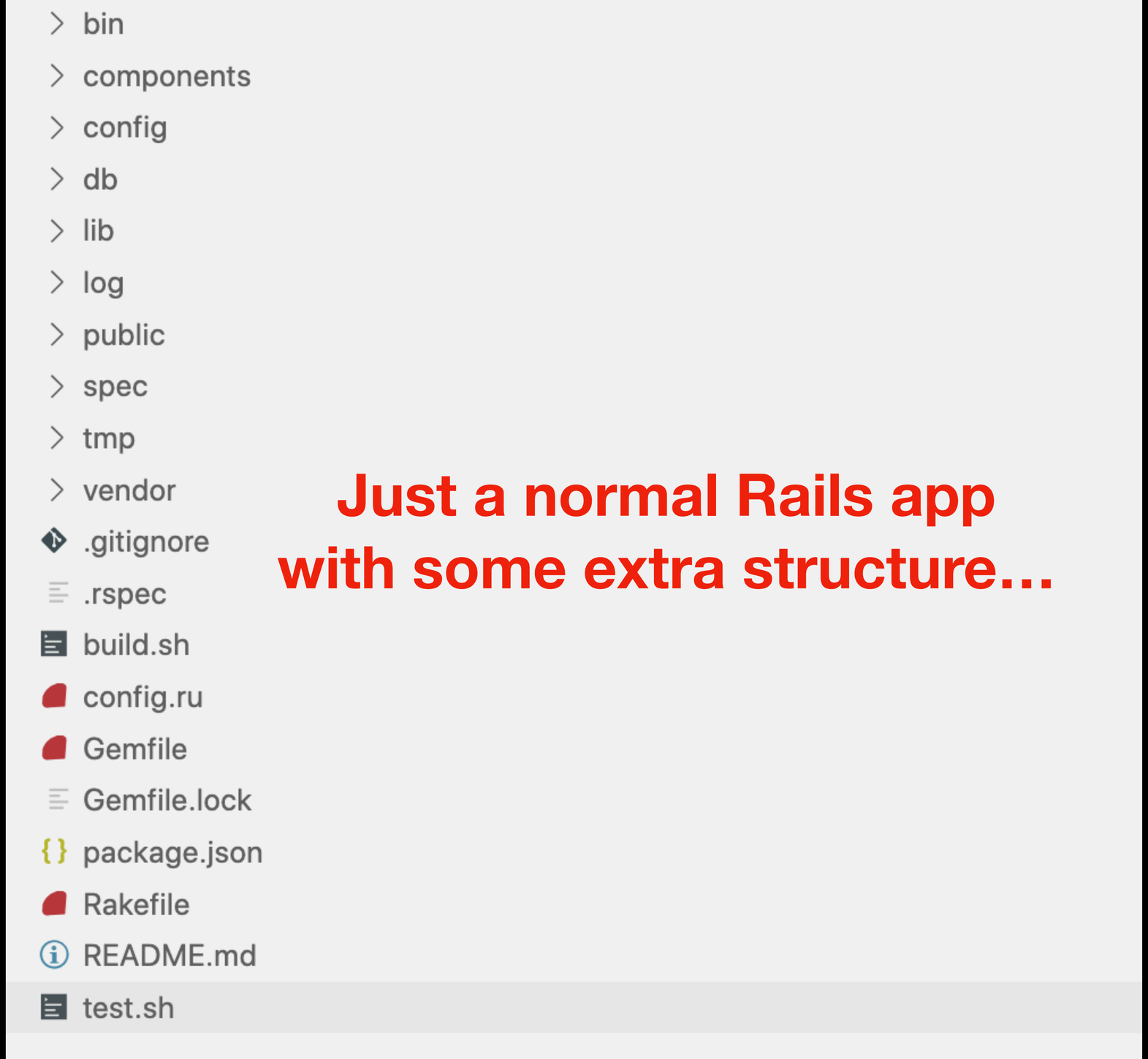
```ruby
require "saulabs/trueskill"

module Predictor
  require "predictor/predictor"
  require "predictor/prediction"
end
```



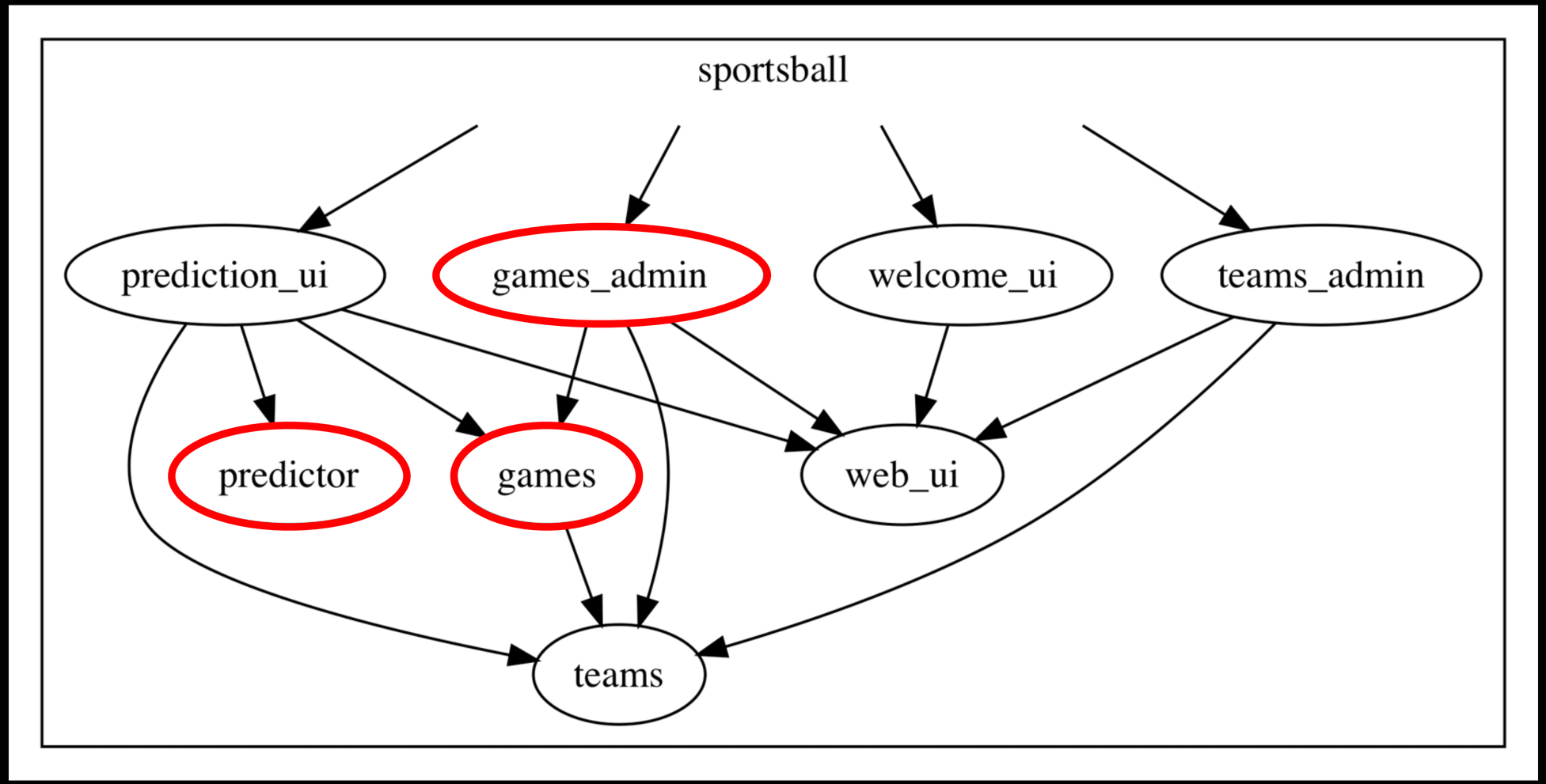

```
#!/bin/bash

exit_code=0

echo "
**********************************************
*** Running predictor gem specs
**********************************************

export BUNDLE_GEMFILE=`pwd`/Gemfile
bundle install | grep Installing
bundle exec rspec spec
((exit_code+=$?))

exit $exit_code
```




```
#!/bin/bash

result=0

cd "$( dirname "${BASH_SOURCE[0]}" )"

for test_script in $(find . -name test.sh | sort); do
  pushd `dirname $test_script` > /dev/null
  ./test.sh
  ((result+=$?))
  popd > /dev/null
done

if [ $result -eq 0 ]; then
  echo "SUCCESS"
else
  echo "FAILURE"
fi

exit $result
```



Just a normal Rails app with some extra structure…


sportsball
prediction_ui
games_admin
welcome_ui
teams_admin
predictor
games
web_ui
teams

Structural Costs + Benefits

```ruby
module Predictor
  class Predictor
    def initialize(teams)
      @teams_hash = teams.inject({}) do |memo, team|
        memo[team.id] = {
            team: team,
            rating: [Saulabs::TrueSkill::Rating.new(1500.0, 1000.0, 1.0)]
        }
        memo
      end
    end

    def learn(games)
      games.each do |game|
        rating0 = @teams_hash[game.first_team_id][:rating]
        rating1 = @teams_hash[game.second_team_id][:rating]
        game_result = game.winning_team == 1 ? [rating0, rating1] : [rating1, rating0]

        Saulabs::TrueSkill::FactorGraph.new(game_result, [1, 2]).update_skills
      end
    end

    def predict(first_team, second_team)
      team1 = @teams_hash[first_team.id][:team]
      team2 = @teams_hash[second_team.id][:team]
      winner = would_first_team_win?(first_team, second_team) ? team1 : team2

      ::Predictor::Prediction.new(team1, team2, winner)
    end

    def would_first_team_win?(team0, team1)
      @teams_hash[team0.id][:rating].first.mean > @teams_hash[team1.id][:rating].first.mean
    end
  end
end
```

```ruby
module Predictor
  class Predictor
    def initialize(teams)
      @teams_hash = teams.inject({}) do |memo, team|
        memo[team.id] = {
            team: team,
            rating: [Saulabs::TrueSkill::Rating.new(1500.0, 1000.0, 1.0)]
        }
        memo
      end
    end

    def learn(games)
      games.each do |game|
        rating0 = @teams_hash[game.first_team_id][:rating]
        rating1 = @teams_hash[game.second_team_id][:rating]
        game_result = game.winning_team == 1 ? [rating0, rating1] : [rating1, rating0]

        Saulabs::TrueSkill::FactorGraph.new(game_result, [1, 2]).update_skills
      end
    end

    def predict(first_team, second_team)
      team1 = @teams_hash[first_team.id][:team]
      team2 = @teams_hash[second_team.id][:team]
      winner = would_first_team_win?(first_team, second_team) ? team1 : team2

      ::Predictor::Prediction.new(team1, team2, winner)
    end

    def would_first_team_win?(team0, team1)
      @teams_hash[team0.id][:rating].first.mean > @teams_hash[team1.id][:rating].first.mean
    end
  end
end
```

Method

Costs

2 LOC

Benefits

Allow Reuse

Hide logic

```ruby
module Predictor
  class Predictor
    def initialize(teams)
      @teams_hash = teams.inject({}) do |memo, team|
        memo[team.id] = {
            team: team,
            rating: [Saulabs::TrueSkill::Rating.new(1500.0, 1000.0, 1.0)]
        }
        memo
      end
    end

    def learn(games)
      games.each do |game|
        rating0 = @teams_hash[game.first_team_id][:rating]
        rating1 = @teams_hash[game.second_team_id][:rating]
        game_result = game.winning_team == 1 ? [rating0, rating1] : [rating1, rating0]

        Saulabs::TrueSkill::FactorGraph.new(game_result, [1, 2]).update_skills
      end
    end

    def predict(first_team, second_team)
      team1 = @teams_hash[first_team.id][:team]
      team2 = @teams_hash[second_team.id][:team]
      winner = would_first_team_win?(first_team, second_team) ? team1 : team2

      ::Predictor::Prediction.new(team1, team2, winner)
    end

    def would_first_team_win?(team0, team1)
      @teams_hash[team0.id][:rating].first.mean > @teams_hash[team1.id][:rating].first.mean
    end
  end
end
```

Class

Costs

2 LOC

Benefits

Bundle logic and data

Allow Reuse

```ruby
module Predictor
  class Predictor
    def initialize(teams)
      @teams_hash = teams.inject({}) do |memo, team|
        memo[team.id] = {
            team: team,
            rating: [Saulabs::TrueSkill::Rating.new(1500.0, 1000.0, 1.0)]
        }
        memo
      end
    end

    def learn(games)
      games.each do |game|
        rating0 = @teams_hash[game.first_team_id][:rating]
        rating1 = @teams_hash[game.second_team_id][:rating]
        game_result = game.winning_team == 1 ? [rating0, rating1] : [rating1, rating0]

        Saulabs::TrueSkill::FactorGraph.new(game_result, [1, 2]).update_skills
      end
    end

    def predict(first_team, second_team)
      team1 = @teams_hash[first_team.id][:team]
      team2 = @teams_hash[second_team.id][:team]
      winner = would_first_team_win?(first_team, second_team) ? team1 : team2

      ::Predictor::Prediction.new(team1, team2, winner)
    end

    def would_first_team_win?(team0, team1)
      @teams_hash[team0.id][:rating].first.mean > @teams_hash[team1.id][:rating].first.mean
    end
  end
end
```

Namespace

Costs

2 LOC

Benefits

Prevent naming collisions in the global namespace

Group classes and methods

Component

```ruby
module Predictor
  class Predictor
    def initialize(teams)
      @teams_hash = teams.inject({}) do |memo, team|
        memo[team.id] = {
            team: team,
            rating: [Saulabs::TrueSkill::Rating.new(1500.0, 1000.0, 1.0)]
        }
        memo
      end
    end

    def learn(games)
      games.each do |game|
        rating0 = @teams_hash[game.first_team_id][:rating]
        rating1 = @teams_hash[game.second_team_id][:rating]
        game_result = game.winning_team == 1 ? [rating0, rating1] : [rating1, rating0]

        Saulabs::TrueSkill::FactorGraph.new(game_result, [1, 2]).update_skills
      end
    end

    def predict(first_team, second_team)
      team1 = @teams_hash[first_team.id][:team]
      team2 = @teams_hash[second_team.id][:team]
      winner = would_first_team_win?(first_team, second_team) ? team1 : team2

      ::Predictor::Prediction.new(team1, team2, winner)
    end

    def would_first_team_win?(team0, team1)
      @teams_hash[team0.id][:rating].first.mean > @teams_hash[team1.id][:rating].first.mean
    end
  end
end
```

WHAT IS *A* COMPONENT?

Labeled Content

+

Explicit Dependencies

Labeled, Unique Content

+

Explicit, Directed, Acyclic Dependencies


?
Bunnahabhain
AUCHENTOSHAN
"AS WE GET IT"
105° PROOF
CADENHEAD'S
COLLECTION
ABERLOUR
a'bunadh
TOBERMORY
SINGLE MALT
SCOTCH WHISKY


Chex
Kix
Cheerios
Total
Whiskas
Friskies
Fresh Step
Ultra Pearls
Kit 'N Kaboodle
Original Choice
Cat Food

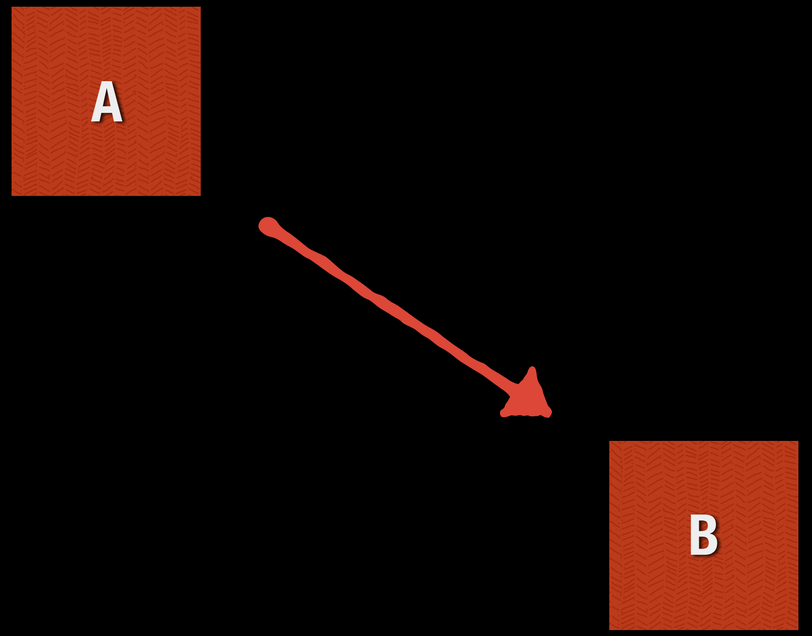

Kids' Papers
Memories
China
Cords
Blankets
Decor
Decor
Decor
School Items
Decor
Decor

Labeled, Unique Content

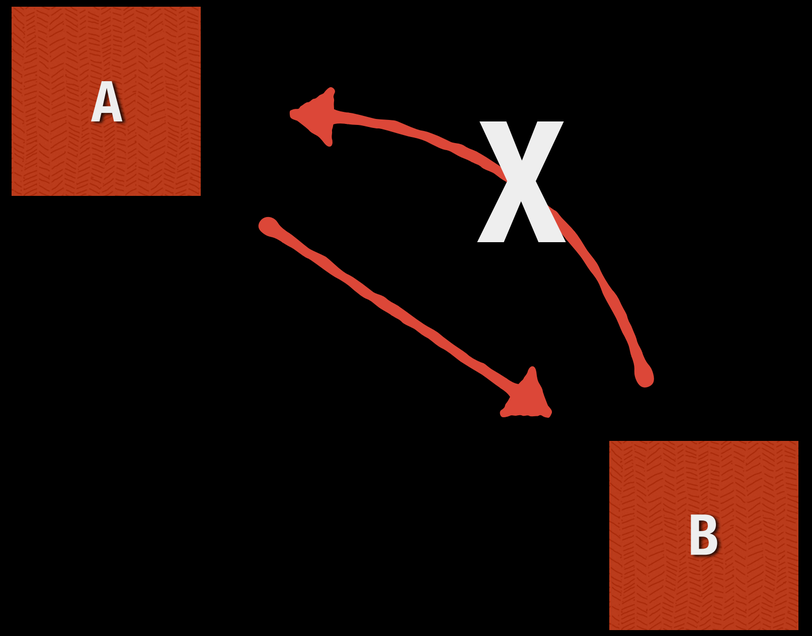
+

Explicit, Directed, Acyclic Dependencies

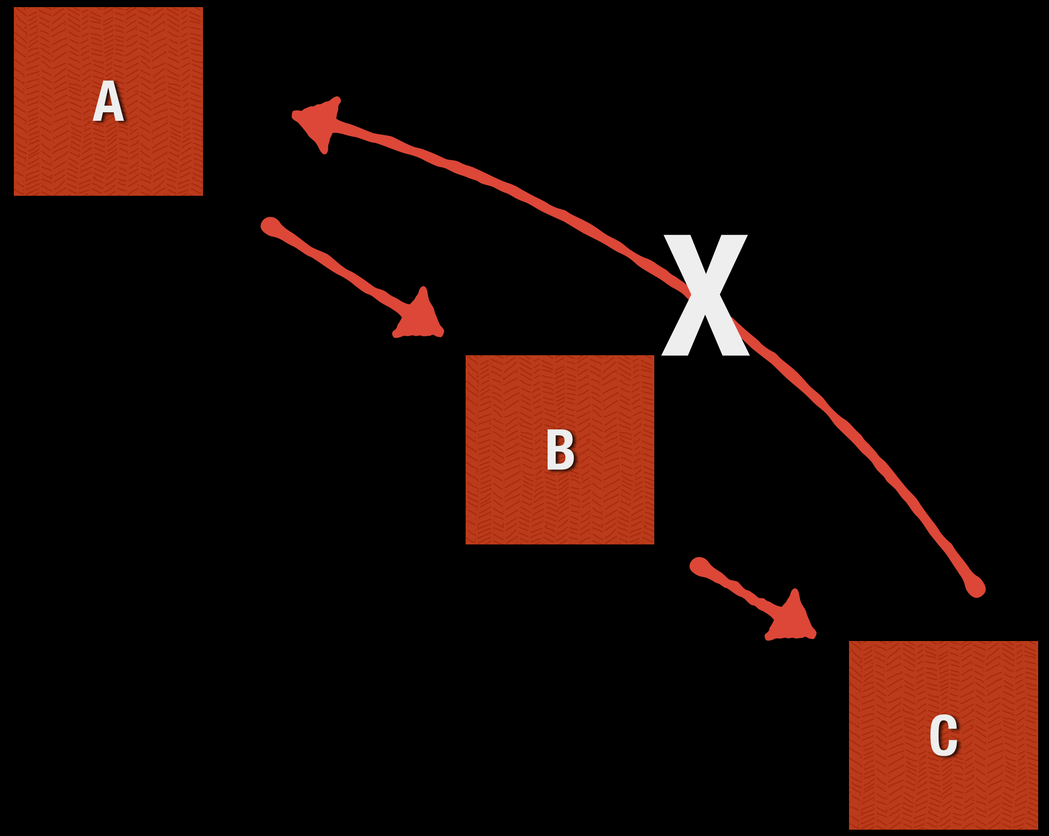
A depends on B …

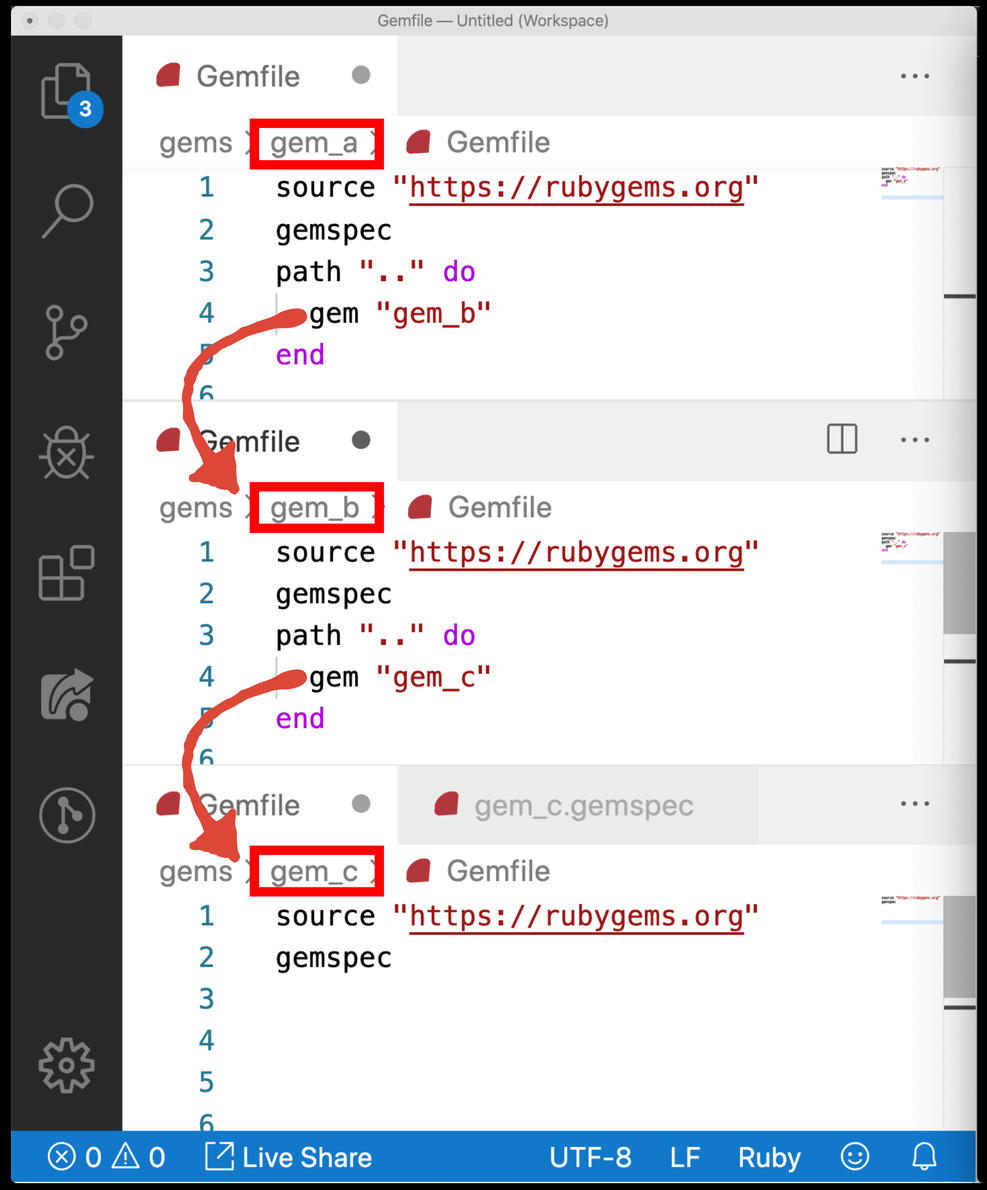


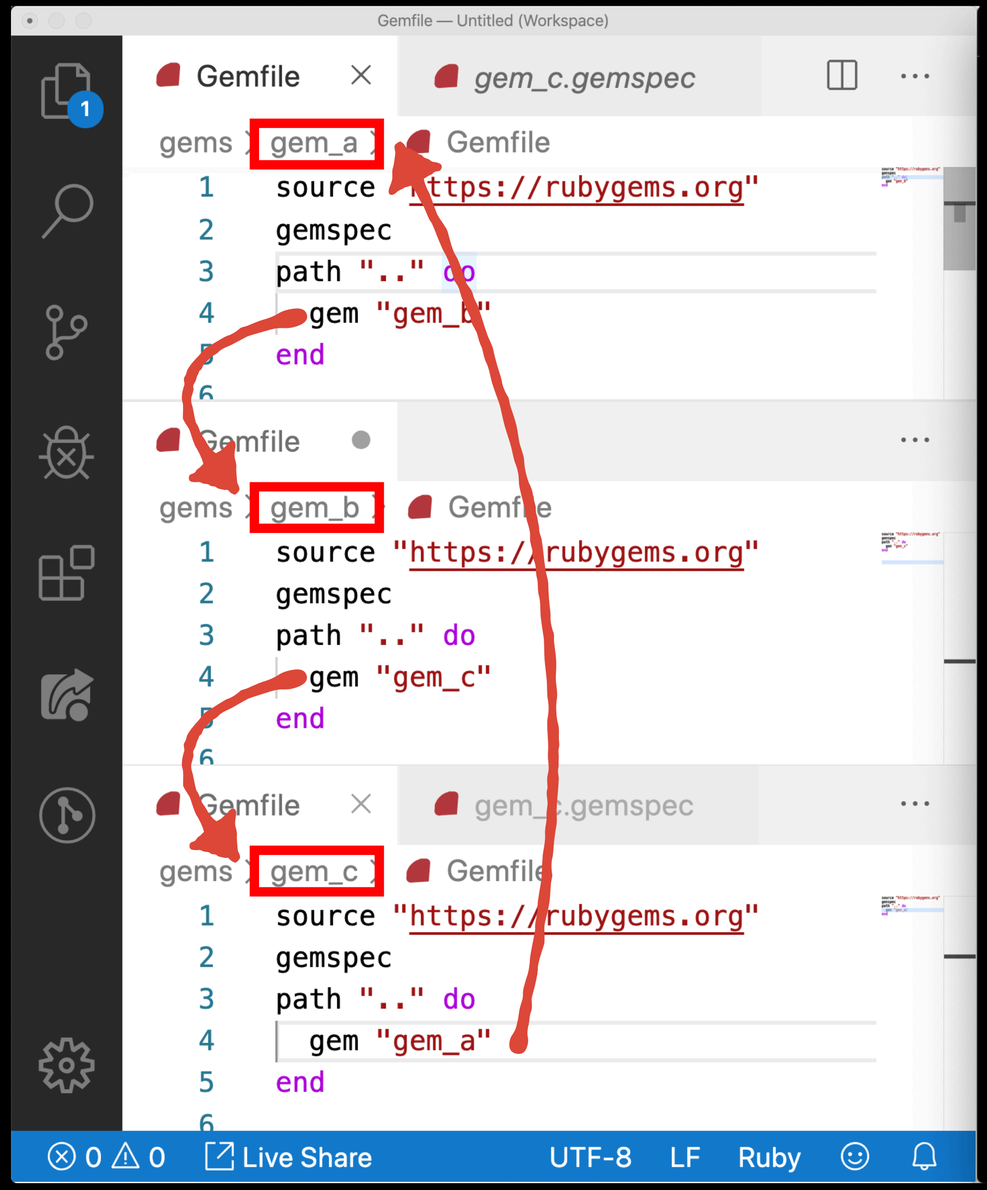
thus B can not depend on A



… and you can’t cheat




Gemfile — Untitled (Workspace)
Gemfile
gems
gem_a
Gemfile
1 source "https://rubygems.org"
2 gemspec
3 path ".." do
4 gem "gem_b"
5 end
Gemfile
gems
gem_b
Gemfile
1 source "https://rubygems.org"
2 gemspec
3 path ".." do
4 gem "gem_c"
5 end
Gemfile
gem_c.gemspec
gems
gem_c
Gemfile
1 source "https://rubygems.org"
2 gemspec
0 0
Live Share
UTF-8
LF
Ruby


Gemfile — Untitled (Workspace)
Gemfile
gem_c.gemspec
gems
gem_a
Gemfile
1 source "https://rubygems.org"
2 gemspec
3 path ".." do
4 gem "gem_b"
5 end
Gemfile
gems
gem_b
Gemfile
1 source "https://rubygems.org"
2 gemspec
3 path ".." do
4 gem "gem_c"
5 end
Gemfile
gem_c.gemspec
gems
gem_c
Gemfile
1 source "https://rubygems.org"
2 gemspec
3 path ".." do
4 gem "gem_a"
5 end
0 0
Live Share
UTF-8
LF
Ruby

Structure	Benefit	Cost
Method	Allow reuse, hide logic	2
Class	Bundle logic and data, allow reuse	2
Namespace	Prevent naming collisions, group classes and methods	2

Gem

```ruby
module Predictor
  class Predictor
    def initialize(teams)
      @teams_hash = teams.inject({}) do |memo, team|
        memo[team.id] = {
            team: team,
            rating: [Saulabs::TrueSkill::Rating.new(1500.0, 1000.0, 1.0)]
        }
        memo
      end
    end

    def learn(games)
      games.each do |game|
        rating0 = @teams_hash[game.first_team_id][:rating]
        rating1 = @teams_hash[game.second_team_id][:rating]
        game_result = game.winning_team == 1 ? [rating0, rating1] : [rating1, rating0]

        Saulabs::TrueSkill::FactorGraph.new(game_result, [1, 2]).update_skills
      end
    end

    def predict(first_team, second_team)
      team1 = @teams_hash[first_team.id][:team]
      team2 = @teams_hash[second_team.id][:team]
      winner = would_first_team_win?(first_team, second_team) ? team1 : team2

      ::Predictor::Prediction.new(team1, team2, winner)
    end

    def would_first_team_win?(team0, team1)
      @teams_hash[team0.id][:rating].first.mean > @teams_hash[team1.id][:rating].first.mean
    end
  end
end
```

bundle gem awesome_lib

```
(base) ➜  awesome_lib git:(master) ✗ cloc .
      12 text files.
      12 unique files.
       3 files ignored.

github.com/AlDanial/cloc v 1.84  T=0.02 s (581.3 files/s, 7614.5 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Ruby                             7             16              8             57
Markdown                         1             16              0             19
YAML                             1              0              0              7
Bourne Again Shell               1              2              1              5
-------------------------------------------------------------------------------
SUM:                            10             34              9             88
-------------------------------------------------------------------------------
```

Gem

Costs

88 LOC

Benefits

Allow Distribution

Create provably independent package of code

rails plugin new components/awesome_engine --full --mountable

```
(base) ➜  awesome_engine git:(master) ✗ cloc .
      67 text files.
      66 unique files.
      18 files ignored.

github.com/AlDanial/cloc v 1.84  T=0.10 s (591.2 files/s, 9860.5 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Ruby                            46            123            245            259
HTML                             3             15              3            182
ERB                              4              6              0             37
YAML                             4             11             62             29
Markdown                         1              7              0             21
JavaScript                       3              0             19              0
CSS                              1              0             15              0
-------------------------------------------------------------------------------
SUM:                            62            162            344            528
-------------------------------------------------------------------------------
```

Gem + Engine

Costs

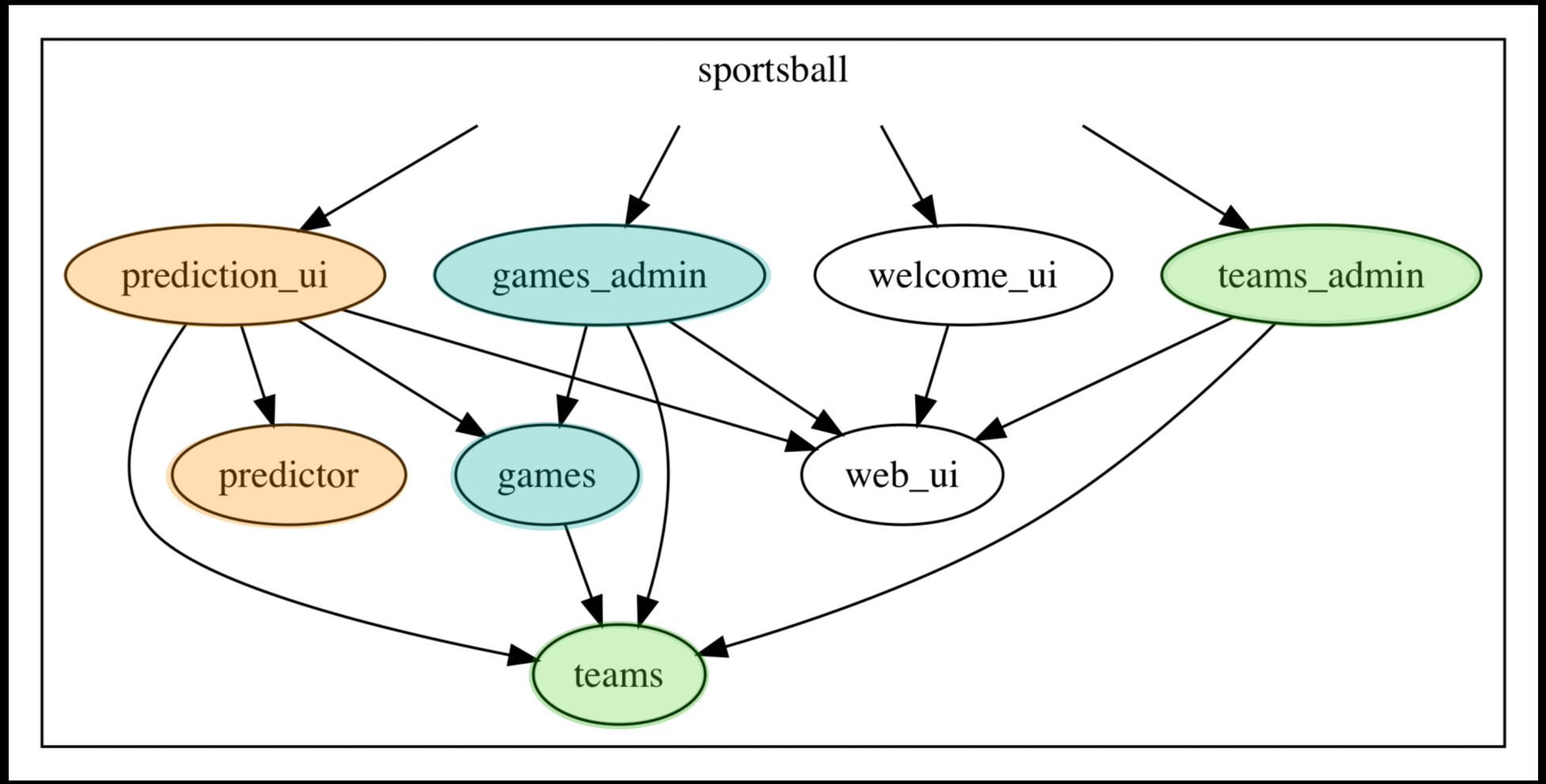
528 LOC

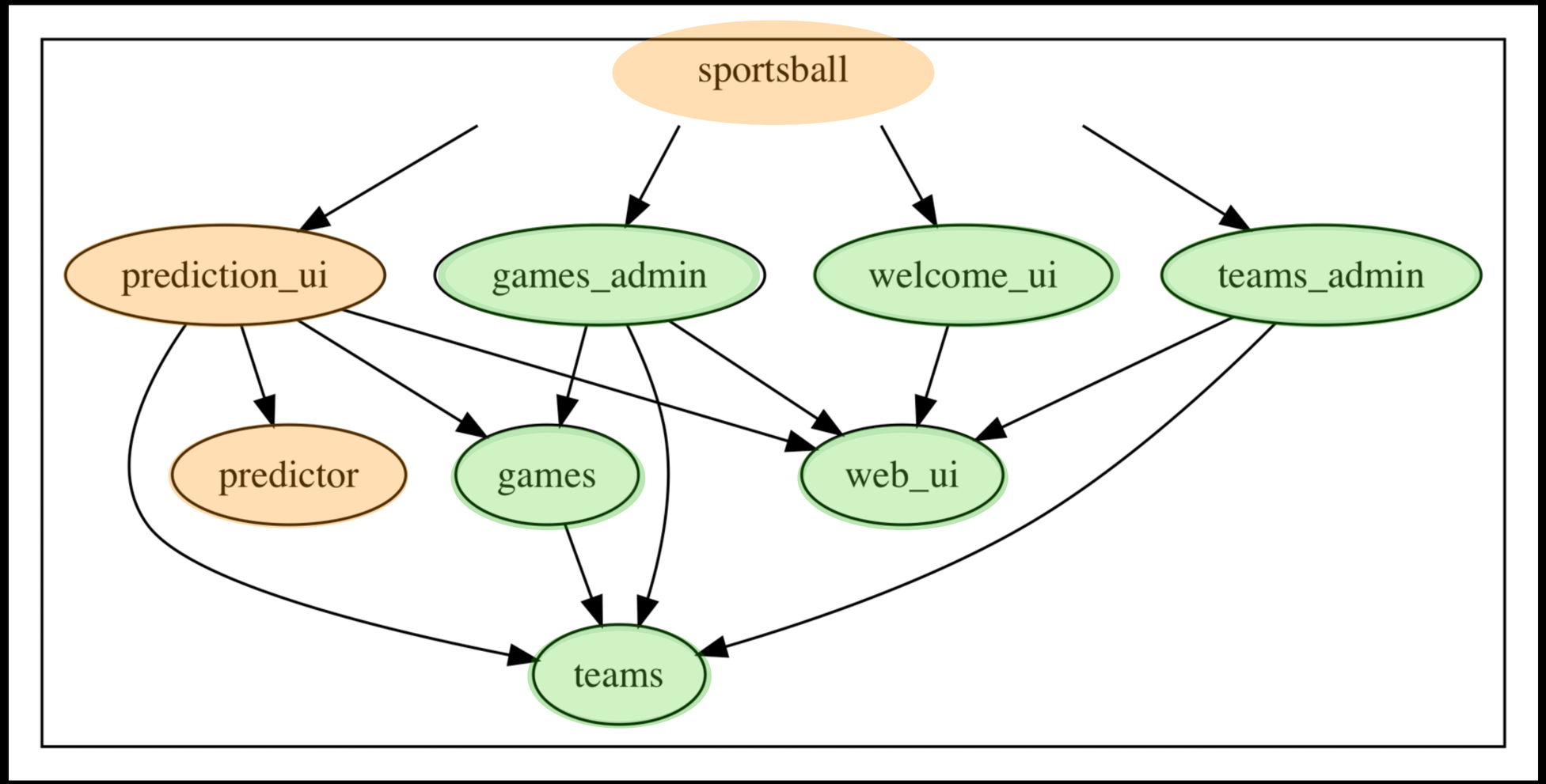
Benefits

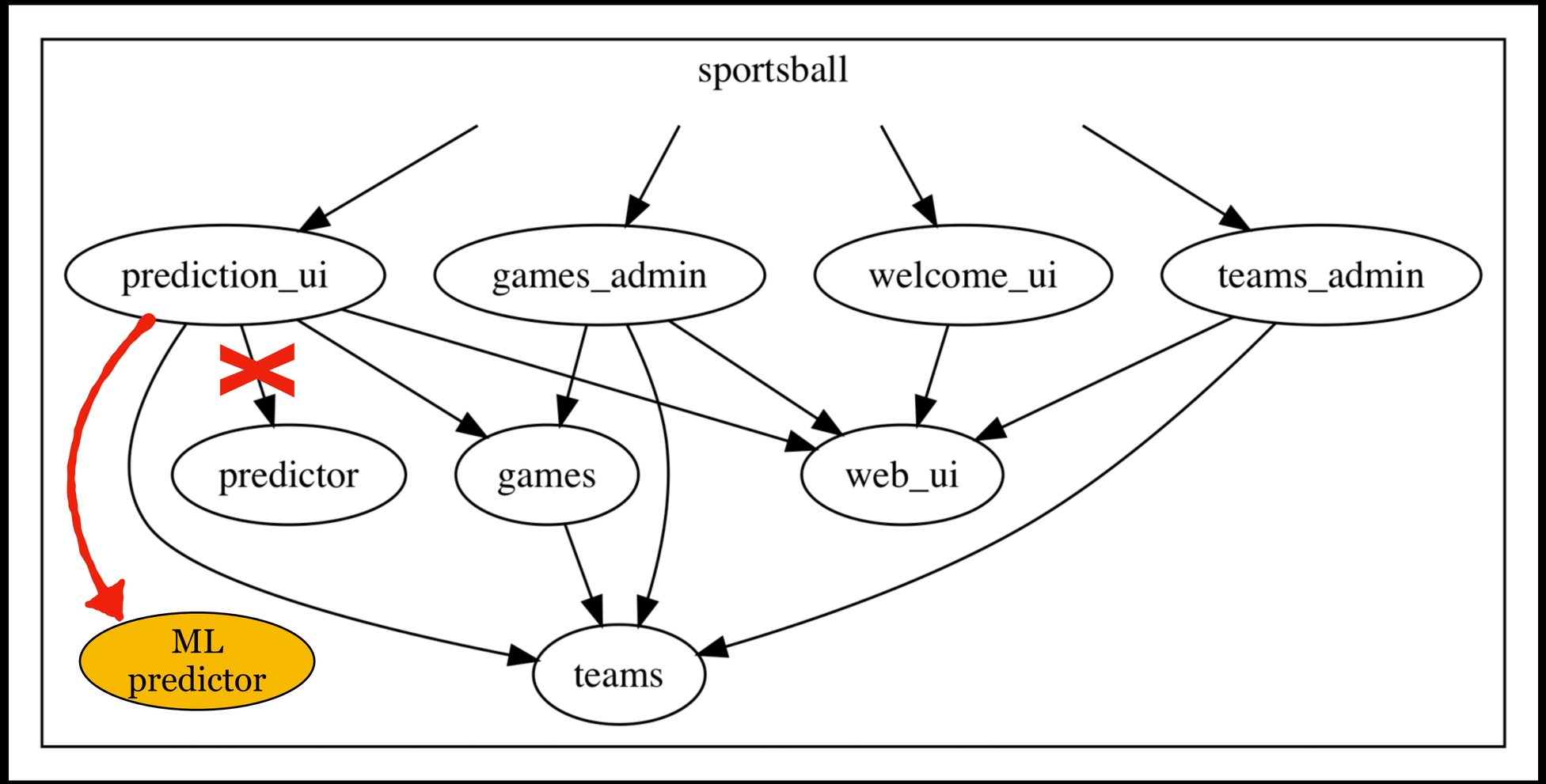
Allow Distribution

Create provably independent package of code **with Rails**

Structure	Benefit	Cost
Method	Allow reuse, hide logic	2
Class	Bundle logic and data, allow reuse	2
Namespace	Prevent naming collisions, group classes and methods	2
Component	Allow Distribution, Create provably independent package of code	88
Rails Component	Allow Distribution, Create provably independent package of code with Rails	528


sportsball
prediction_ui
games_admin
welcome_ui
teams_admin
predictor
games
web_ui
teams


sportsball
prediction_ui
games_admin
welcome_ui
teams_admin
predictor
games
web_ui
teams


sportsball
prediction_ui
games_admin
welcome_ui
teams_admin
predictor
games
web_ui
ML
predictor
teams

10 component with 10 elements	10,130
5 component with 20 elements	5,242,775
4 component with 25 elements	134,217,624
2 component with 50 elements	2,251,799,813,685,150
1 component with 100 elements	1,267,650,600,228,230,000,000,000,000,000

* assuming a lot of stuff, including complexity growing according to Reed’s law

Structure	Benefit	Cost
Method	Allow reuse, hide logic	2
Class	Bundle logic and data, allow reuse	2
Namespace	Prevent naming collisions, group classes and methods	2
Component	Allow Distribution, Create provably independent package of code	88
Rails Component	Allow Distribution, Create provably independent package of code with Rails	528

```jsx
const ComponentA = (props) => {
  if (props.x == 0) return null
  console.log("component a", props.x)
  return <ComponentB x={props.x - 1}></ComponentB>
}

const ComponentB = (props) => {
  if (props.x == 0) return null
  console.log("component b", props.x)
  return <ComponentA x={props.x - 1}></ComponentA>
}
```

```
component a 10
component b 9
component a 8
component b 7
component a 6
component b 5
component a 4
component b 3
component a 2
component b 1
```

ComponentB.tsx

src > ComponentB.tsx > ...

```tsx
import React, { } from 'react';
import ComponentA from './ComponentA';

const ComponentB = (props) => {
    if (props.x == 0) return null
    console.log("component b", props.x)
    return <ComponentA x={props.x - 1}></ComponentA>
}

export default ComponentB;
```

ComponentA.tsx

src > ComponentA.tsx > ...

```tsx
import React, { } from 'react';
import ComponentB from './ComponentB';

const ComponentA = (props) => {
    if (props.x == 0) return null
    console.log("component b", props.x)
    return <ComponentB x={props.x - 1}></ComponentB>
}

export default ComponentA;
```

Costs

2 LOC

```
component a 10
component b 9
component a 8
component b 7
component a 6
component b 5
component a 4
component b 3
component a 2
component b 1

Require cycle: src/ComponentA.tsx -> src/ComponentB.tsx -> src/ComponentA.tsx

Require cycles are allowed, but can result in uninitialized values. Consider re
- node_modules/expo/build/environment/muteWarnings.fx.js:18:23 in warn
- node_modules/metro/src/lib/polyfills/require.js:115:8 in metroRequire
* src/ComponentB.tsx:2:0 in <unknown>
```

Structure	Benefit	Cost
Method	Allow reuse, hide logic	2
Class	Bundle logic and data, allow reuse	2
Namespace	Prevent naming collisions, group classes and methods	2
Component	Allow Distribution, Create provably independent package of code	88
Rails Component	Allow Distribution, Create provably independent package of code with Rails	528

Let’s make it better!

Share the stories
of your large app!

Reduce Rails component costs!

Improve CI/CD integrations

Fix cross-component dependency versioning

Can we create component extractors?


an Schopohl Photography


Dreieck
Hannover-Nord
7
7