

# Testing Rails

**Without the Rails environment**

Joe Corcoran • [corcoran.io](http://corcoran.io) • [@josephcorcoran](https://twitter.com/josephcorcoran)

**Why?**

# Why?

- **Fast development feedback loop**

# Why?

- **Fast development feedback loop**
- **Highlights design problems**

# Why?

- **Fast development feedback loop**
- **Highlights design problems**
- **Enforces discipline**

# Model

```
class User < ActiveRecord::Base
  has_many :widgets

  def say_hello
    Rails.logger.info("Hello from #{name}")
  end

  def add_widget(widget)
    widgets << widget
  end
end
```

# Model spec

```
require 'rails_helper'
```

```
RSpec.describe User, type: :model do  
  let(:user) { User.new(name: 'Joe') }
```

```
  describe '#say_hello' do  
    it 'says hello' do  
      expect(Rails.logger).to receive(:info).with('Hello from Joe')  
      user.say_hello  
    end  
  end
```

```
  describe '#add_widget' do  
    it 'adds widget to user' do  
      widget = Widget.new(name: 'Thingy')  
      user.add_widget(widget)  
      expect(user.widgets).to include widget  
    end  
  end  
end
```

# Model spec without Rails

```
require 'spec_helper'
```

```
require_relative '../..../app/models/user'
```

```
require_relative '../..../app/models/widget'
```

```
RSpec.describe User, type: [:model, :db] do
```

```
  let(:user) { User.new(name: 'Joe') }
```

```
    # ...
```

```
end
```



# Model spec without Rails

```
require 'active_record'
```

```
class User < ActiveRecord::Base
```

```
  # ...
```

```
end
```

## Extra setup

```
module Rails
  def self.logger
    @logger ||= begin
      require 'stringio'
      Logger.new(StringIO.new)
    end
  end
end
end unless defined?(Rails)
```

# Extra setup

```
config.before(type: :db) do
  config = YAML.load(File.read('config/database.yml'))
  ActiveRecord::Base.establish_connection(config['test'])
end
```

# Controller

```
class UsersController < ApplicationController
  def index
    @users = User.all
    @widgets = Widget.all
  end

  def new
    head :unauthorized
  end
end
```

# Controller spec

```
require 'rails_helper'
```

```
RSpec.describe UsersController, type: :controller do
```

```
  describe 'GET index' do
```

```
    it 'is successful' do
```

```
      get :index
```

```
      expect(response).to be_ok
```

```
    end
```

```
  end
```

```
  describe 'GET new' do
```

```
    it 'is not accessible' do
```

```
      get :new
```

```
      expect(response).to be_unauthorized
```

```
    end
```

```
  end
```

```
end
```

# Controller spec without Rails

```
require 'spec_helper'

require_relative '../..../app/controllers/users_controller'

RSpec.describe UsersController, type: :rack do
  include Rack::Test::Methods

  describe 'GET index' do
    let(:app) { UsersController.action(:index) }

    it 'is successful' do
      get :index
      expect(last_response).to be_ok
    end
  end

  # ...
end
```

# Controller spec without Rails

```
require_relative './application_controller'
```

```
require_relative '../models/user'
```

```
require_relative '../models/widget'
```

```
class UsersController < ApplicationController
```

```
  # ...
```

```
end
```

## Extra setup

```
config.before(type: :rack) do |example|  
  require 'rack/test'  
  described_class.append_view_path('app/views')  
end
```



```
~/Projects/betamax $ bundle exec guard
```

```
15:43:55 - Guard::RSpec is running
```

```
15:43:55 - Guard is now watching at '~/Projects/betamax'
```

```
15:44:01 - Running: spec/models/user_spec.rb
```

**Is it easy?**

Is it easy?

— No

Is it easy?

— No, because Rails

# Is it easy?

- **No, because Rails**
- **Much easier in a new project**

# Is it easy?

- **No, because Rails**
- **Much easier in a new project**
- **Achievable in small steps**

# Great

**Are there any downsides?**

# Are there any downsides?

# Not possible

expect(assigns[:users]).to include user



# Are there any downsides?

```
class UsersController < ApplicationController
  # ...

  def users
    @users ||= User.all
  end
end

# Not possible
allow(controller).to receive(:users) { double }
```

# Are there any downsides?

```
require_relative '../models/user'
```

```
require_relative '../models/widget'
```

```
require_relative '../../app/controllers/users_controller'
```

# Are there any downsides?

```
require 'active_record'
```

```
Dir['app/models/**/*.rb'].each do |p|  
  require_relative(File.join('..../..', p))  
end
```

**Thanks!**

**This talk**

**[github.com/joecorcoran/talks/tree/master/noenv](https://github.com/joecorcoran/talks/tree/master/noenv)**

**Sample Rails application**

**[github.com/joecorcoran/noenv](https://github.com/joecorcoran/noenv)**