

Slides:

amcaplan.ninja/railsconf2016

GitHub Repos:

[amcaplan/feedback](https://github.com/amcaplan/feedback)

[amcaplan/feedback_scripts](https://github.com/amcaplan/feedback_scripts)

Understanding, Building, and Integrating Rails Engines



**THIS IS A
WORKSHOP**

Assumptions

You are comfortable with:

- Git and Github
- Ruby
- Rails

and can get them all working locally

Ruby 2.3.1, Rails 4.2.6 (i.e. latest stable versions)

What Are Rails Engines?

Engines can be considered miniature applications that provide functionality to their host applications.

– *RailsGuides: Getting Started with Engines*
(<http://guides.rubyonrails.org/engines.html>)

What Are Rails Engines?

Engines let you build gems with access to autoloading magic, and the full Rails MVC architecture.

- *me*

Are Rails Engines Useful?



Are Rails Engines Useful?



jquery-rails



turbolinks

bootstrap-sass

resque-web

rollout-ui

PaperTrail

RailsAdmin

coffee-rails

If Rails is an
MVC
framework...

Can We Build a
Rails Application
without
MODELS?

```
class FooController < ApplicationController
  def index
    @foo_results = FooAPI.call(params)
    render "results"
  end
end
```


Can We Build a
Rails Application
without
MODELS?

Can We Build a
Rails Application
without
MODELS?

Can We Build a
Rails Application
without
VIEWS?


```
class PostsController < ApplicationController
  def index
    @posts = Post.all
    render json: @posts
  end
end
```

```
class FooController < ApplicationController
  def index
    render json: FooAPI.call(params)
  end
end
```

```
class FooController < ApplicationController
  def index
    render text: "foo bar"
  end
end
```


Can We Build a
Rails Application
without
VIEWS?

Can We Build a
Rails Application
without
VIEWS?

Can We Build a
Rails Application
without
CONTROLLERS?

Rack: a Ruby Webserver Interface



Rack provides a minimal interface between webserver that support Ruby and Ruby frameworks.

To use Rack, provide an "app": an object that responds to the `call` method, taking the environment hash as a parameter, and returning an Array with three elements:

- The HTTP response code
- A Hash of headers
- The response body, which must respond to `each`

You can handle an app directly:

```
1 | # my_rack_app.rb
2 |
3 | require 'rack'
4 |
5 | app = Proc.new do |env|
6 |   ['200', {'Content-Type' => 'text/html'}, ['A barebones rack app.']]
7 | end
8 |
9 | Rack::Handler::WEBrick.run app
```

Or, you can use the `rackup` command line tool and avoid specifying details like port and server until runtime:

```
1 | # config.ru
2 |
3 | run Proc.new { |env| ['200', {'Content-Type' => 'text/html'}, ['get rack\'d']] }
```

Invoked like so:

```
$ rackup config.ru
```

... and you're good to go!

Outrageous.

Rack provides a minimal interface between web servers that support Ruby and Ruby frameworks.

To use Rack, provide an "**app**": an object that responds to the **call** method, taking the **environment** hash as a parameter, and **returning an Array** with three elements:

- The HTTP **response code**
- A Hash of **headers**
- The **response body**, which must respond to each

```
require 'rack'
```

```
app = Proc.new do |env|
```

```
  [
```

```
    '200',
```

```
    {'Content-Type' => 'text/html'},
```

```
    ['A barebones rack app.']
```

```
  ]
```

```
end
```

```
Rack::Handler::WEBrick.run app
```

Can We Build a
Rails Application
without
CONTROLLERS?


Can We Build a
Rails Application
without
CONTROLLERS?

**The routes file
matches
HTTP requests
to
Rack Applications**

**You can mount any
Rack Application
inside your
Rails Application**

**YO DAWG, YOU CAN PUT A RACK APPLICATION
IN YOUR RAILS APPLICATION**

**SO SINATRA CAN TAKE THE STAGE
WHILE YOU RIDE RUBY ON RAILS**

Sidekiq Pro idle

DashboardBusyQueuesRetriesScheduledDeadBatches

Stop Polling

2,278,255
Processed

174,505
Failed

0
Busy

0
Enqueued

0
Retries

0
Scheduled

0
Dead


Processes

Quiet AllStop All

Name	Started	Threads	Busy	
ip-10-53-3-189:2281 platform-api-etl reliable Queues: default, professional, facility, places, reference, professional_review, facility_review, maintenance	2 days ago	3	0	<div>QuietStop</div>
ip-10-53-3-189:2658 platform-api-etl reliable Queues: default, professional, facility, places, reference, professional_review, facility_review, maintenance	2 days ago	3	0	<div>QuietStop</div>
ip-10-53-3-189:3041 platform-api-etl reliable Queues: default, professional, facility, places, reference, professional_review, facility_review, maintenance	2 days ago	3	0	<div>QuietStop</div>
ip-10-53-3-189:3424 platform-api-etl reliable Queues: default, professional, facility, places, reference, professional_review, facility_review, maintenance	2 days ago	3	0	<div>QuietStop</div>
ip-10-53-3-189:4207 platform-api-etl reliable Queues: cost	2 days ago	3	0	<div>QuietStop</div>
ip-10-53-3-189:4584 platform-api-etl reliable Queues: cost	2 days ago	3	0	<div>QuietStop</div>
ip-10-53-3-189:4967 platform-api-etl reliable Queues: cost	2 days ago	3	0	<div>QuietStop</div>

Sidekiq v4.1.0 / Sidekiq Pro v3.0.6redis://ip-10-53-3-189.ec2.internal:6379/0#platform_etl_qa20:24:12 UTC

Sidekiq Web

Sidekiq Pro  idle

Dashboard

Busy

Queues

Retries

Scheduled

Dead

Batches

Stop Polling

2,278,255
Processed

174,505
Failed

0
Busy

0
Enqueued

0
Retries

0
Scheduled

0
Dead

Processes

Quiet All

Stop All

Name	Started	Threads	Busy	
ip-10-53-3-189:2281 platform-api-etl reliable Queues: default, professional, facility, places, reference, professional_review, facility_review, maintenance	2 days ago	3	0	<div>Quiet</div> <div>Stop</div>
ip-10-53-3-189:2658 platform-api-etl reliable Queues: default, professional, facility, places, reference, professional_review, facility_review, maintenance	2 days ago	3	0	<div>Quiet</div> <div>Stop</div>
ip-10-53-3-189:3041 platform-api-etl reliable Queues: default, professional, facility, places, reference, professional_review, facility_review, maintenance	2 days ago	3	0	<div>Quiet</div> <div>Stop</div>
ip-10-53-3-189:3424 platform-api-etl reliable Queues: default, professional, facility, places, reference, professional_review, facility_review, maintenance	2 days ago	3	0	<div>Quiet</div> <div>Stop</div>
ip-10-53-3-189:4207 platform-api-etl reliable Queues: cost	2 days ago	3	0	<div>Quiet</div> <div>Stop</div>
ip-10-53-3-189:4584 platform-api-etl reliable Queues: cost	2 days ago	3	0	<div>Quiet</div> <div>Stop</div>
ip-10-53-3-189:4967 platform-api-etl reliable Queues: cost	2 days ago	3	0	<div>Quiet</div> <div>Stop</div>

Sidekiq v4.1.0 / Sidekiq Pro v3.0.6redis://ip-10-53-3-189.ec2.internal:6379/0#platform_etl_qa20:24:12 UTC

Sidekiq Web

```
require 'sidekiq/web'  
mount Sidekiq::Web => '/sidekiq'
```


**You can mount any
Rack Application
inside your
Rails Application**

**Rails Engines
are just
Rack Applications**

Rails Engines
are just
Rack Applications
you can mount inside a
Rails Application

Rails Applications
inherit from
Rails Engines

Rails::Application.superclass
=> Rails::Engine

Rails Applications
are just
Rails Engines
plus code to
run independently

Rails Engines
are just
Rails Applications
minus code to
run independently

Rails Engines
are just
Rack Applications
you can mount inside a
Rails Application

Rails Engines
are just
almost-Rails Applications
you can mount inside a
Rails Application

What Are Rails Engines?

Engines can be considered miniature applications that provide functionality to their host applications.

– *RailsGuides: Getting Started with Engines*
(<http://guides.rubyonrails.org/engines.html>)

**Let's Learn
by
Building!**

**Meet a
Practical Need**

**We Write Rails
Applications**

**Do Users Like
Our Products?**

Solicit Feedback

**Solicit Feedback
On Hundreds of
Rails Applications**

Feedback Engine



Feedback Engine

1. Land on a survey page (form)

Feedback Engine

1. Land on a survey page (form)
2. Submit a survey

Feedback Engine

1. Land on a survey page (form)
2. Submit a survey
3. Show thank-you message

Feedback Engine

1. Land on a survey page (form)
2. Submit a survey
3. Show thank-you message



**Let's Start
Building!**

Stage 0: Basic Setup

Expected Structure

```
rails_engines_railsconf_2016
├── feedback
└── feedback_scripts
```

Clone the Project Repo

```
$ git clone git@github.com:amcaplan/  
feedback.git
```

OR if using HTTPS:

```
$ git clone https://github.com/amcaplan/  
feedback.git
```

Clone the Scripts Repo

```
$ git clone git@github.com:amcaplan/  
feedback_scripts.git
```

OR if using HTTPS:

```
$ git clone https://github.com/amcaplan/  
feedback_scripts.git
```

Create a Project

```
$ rails plugin new feedback --mountable  
--skip-test-unit --dummy-path=spec/dummy
```

```
$ cd feedback
```

(If you weren't already in a repo: `$ git init`)

```
$ git add .
```

```
$ git commit -m "Initial Commit"
```

Set Up RSpec

In feedback.gemspec:

```
s.add_development_dependency "rspec-rails", "~> 3.0"
```

```
$ bundle install
```

```
$ rails generate rspec:install
```

In lib/feedback/engine.rb:

```
config.generators do |g|  
  g.test_framework :rspec  
end
```

RSpec Rails Engine Fix

Change line 3 of spec/rails_helper.rb:

```
require File.expand_path(  
  '../..../spec/dummy/config/environment', __FILE__)
```

Commit It!

```
$ rspec
```

```
$ git add .
```

```
$ git commit -m "Setup RSpec"
```


Anything Awry?

If you need to catch up:

```
$ git checkout engine-with-rspec
```

```
$ git checkout -b extra-life
```

Stage 1:

Thank-You Page

Write the Spec

Well, actually just let the script do it:

```
$ ../feedback_scripts/thanks_page_specs
```

Set Up an Endpoint

Modify config/routes.rb:

```
get 'thanks', to: 'surveys#thanks'
```

```
$ rails g controller surveys --no-helper  
--no-controller-specs --no-view-specs
```

Add an action to app/controllers/feedback/surveys:

```
def thanks  
end
```

Add a view: app/views/feedback/surveys/thanks.html.erb
(choose whatever text you want)

Commit It!

```
$ rspec # Just to be sure!
```

```
$ git add .
```

```
$ git commit -m "Add thanks page to our engine"
```

Is Something Amiss?

If you need to catch up:

```
$ git checkout thanks-page
```

```
$ git checkout -b play-again
```

Interlude:
Integration Time!

Include in a New Rails App

```
$ cd ..
```

```
$ rails new host
```

```
$ cd host
```

Update Gemfile:

```
gem 'feedback', path: '../feedback'
```

```
$ bundle install
```


Fill Out the Gemspec

```
$ cd ../feedback
```

In feedback.gemspec:

```
- s.homepage      = "TODO"
- s.summary       = "TODO: Summary of Feedback."
- s.description   = "TODO: Description of Feedback."
+ s.homepage      = "http://railsconf.com"
+ s.summary       = "Summary of Feedback."
+ s.description   = "Description of Feedback."
```

```
$ git add .
```

```
$ git commit -m "Fill out the Gemspec"
```

Mount the Engine

```
$ cd ../host
```

```
$ bundle install # SUCCESS!!!
```

In config/routes.rb:

```
mount Feedback::Engine => '/feedback'
```

Try It Out!

In the host directory:

```
$ rails s
```

In your browser, navigate to

<http://localhost:3000/feedback/thanks>

Stage 2: Storing the Survey Response

Generate a Model

```
$ cd ../feedback
```

```
$ rails g model survey_response approval:boolean
```

```
$ rake db:migrate
```

```
$ git add .
```

```
$ git commit -m "Add Feedback::SurveyResponse Model"
```

**Fill In the
Controller Action**

Write the Spec

Use the Script, Luke!

```
$ ../feedback_scripts/survey_submit_specs
```


Exercise A:

Process the

Survey Response

(7 minutes)

Fill Out the Endpoint

In config/routes.rb:

```
post 'survey_responses', to: 'surveys#create'
```

In app/controllers/feedback/surveys_controller:

```
def create
  Feedback::SurveyResponse.create!(survey_response_params)
  redirect_to thanks_path
end
```

```
private
```

```
def survey_response_params
  params.require(:survey_response).permit(:approval)
end
```

Commit It!

```
$ rspec
```

```
$ git add .
```

```
$ git commit -m "Add code to store submitted survey responses"
```

Got Stuck?

If you need to catch up:

```
$ git checkout form-submission
```

```
$ git checkout -b 1up
```

Stage 3:

Form a Form

Add Capybara

In feedback.gemspec:

```
s.add_development_dependency "capybara", "~> 2.5.0"
```

```
$ bundle install
```

```
$ git add .
```

```
$ git commit -m "Add Capybara for feature testing"
```

Write a Feature Test

Scriptacular!

```
$ ../feedback_scripts/survey_form_specs
```


Exercise B:
Create the
Survey Form
(7 minutes)

Create the Endpoint

In config/routes.rb:

```
get 'survey_responses/new', to: 'surveys#new'
```

In app/controllers/feedback/surveys_controller:

```
def new  
  @survey_response = Feedback::SurveyResponse.new  
end
```

Build the Form

Create app/views/feedback/surveys/new.html.erb:

```
<%= form_for(@survey_response) do |f| %>
  Do you like our website?<br/>
  <%= f.radio_button(:approval, true) %>
  <%= f.label(:approval, 'Yes', value: true) %><br/>
  <%= f.radio_button(:approval, false) %>
  <%= f.label(:approval, 'No', value: false) %><br/>
  <%= f.submit(value: 'Submit') %>
<% end %>
```

If You Like It Then You Should Put a Commit On It

```
$ git add .
```

```
$ git commit -m "Add form to submit the survey"
```

Did Something Go Wrong?

If you want to see the completed engine:

```
$ git checkout complete-engine
```

Try It Out!

```
$ cd ../host  
$ rails s
```

Navigate to

http://localhost:3000/feedback/survey_responses/new

What went wrong???

Engine Migrations

Engine migrations are not automatically run on the host.

We need to copy them over, then run:

```
$ rake feedback:install:migrations  
$ rake db:migrate
```

An Alternative?

```
module Feedback
  class Engine < ::Rails::Engine
    isolate_namespace Feedback

    initializer :append_migrations do |app|
      unless app.root.to_s.match root.to_s
        app.config.paths["db/migrate"] +=
          config.paths["db/migrate"].expanded
      end
    end
  end
end
end
```


**What Cool
Features Could
We Add to this
Engine?**

**What Amazing
Rails Engine
Will YOU Build
Next?**

Ariel Caplan

@amcaplan

Thanks!



Title Image Credit: “The Little Engine That Could” by Cliff:
<https://www.flickr.com/photos/nostri-imago/2851664965/> / CC BY 2.0