🪝 Catch Me If You Can

Consuming Webhooks in Ruby on Rails

👋 @colinloretz and @excid3

👋 Hello! Welcome!

Setting the Stage

Why Webhooks?

Github Repository

https://github.com/colinloretz/railsconf-webhooks

You can take a look at each code addition in the GitHub PRs.

Doing it live

- Step 1: Controllers & Routes
- Step 2: Adding A Model
- Step 3: Background Jobs
- Step 4: Adding Verification
- Step 5: Adding Some Tests

Step 1: Creating our Routes & Controllers

Creating a Webhooks::BaseController

```
rails generate controller Webhooks::BaseController
```

```ruby
# app/controllers/webhooks/base_controller.rb

class Webhooks::BaseController < ApplicationController
  # Disable CSRF checking on webhooks because they do not originate from the browser
  skip_before_action :verify_authenticity_token

  def create
    head :ok
  end
end
```

Creating a Movies Controller

```
rails generate controller Webhooks::MoviesController
```

```
# app/controllers/webhooks/movies_controller.rb

class Webhooks::MoviesController < Webhooks::BaseController
  # A controller for catching new movie webhooks
  #
  # To send a sample webhook locally:
  #
  #   curl -X POST http://localhost:3000/webhooks/movies
  #     -H 'Content-Type: application/json'
  #     -d '{"title":"Titanic"}'
  #
  # If you'd like to override the base controller's behavior, you can do so here
  # def create
  #   head :ok
  # end
end
```

Creating a Stripe Controller

```
rails generate controller Webhooks::StripeController
```

```
# app/controllers/webhooks/stripe_controller.rb

class Webhooks::StripeController < Webhooks::BaseController
  # If you'd like to override the base controller's behavior, you can do so here
  # def create
  #   head :ok
  # end
end
```

```
curl -X POST http://localhost:3000/webhooks/movies
     -H 'Content-Type: application/json'
     -d '{"title":"Dungeons & Dragons: Honor Among Thieves"}'
```

Let's Add Some Routes!

```ruby
# config/routes.rb

Rails.application.routes.draw do
  namespace :webhooks do
    # /webhooks/movies routed to our Webhooks::MoviesController
    resource :movies, controller: :movies, only: [:create]

    # /webhooks/stripe routed to our Webhooks::StripeController
    resource :stripe, controller: :stripe, only: [:create]
  end

  # your other routes here
end
```

Testing Our Routes

```
curl -X POST http://localhost:3000/webhooks/movies
     -H 'Content-Type: application/json'
     -d '{"title":"The Fifth Element"}'
```

Step 2: Adding A Model

```
rails generate model InboundWebhook status:string body:text
```

```
rails db:migrate
```

Updating Our Controller

```ruby
# app/controllers/webhooks/base_controller.rb

class Webhooks::BaseController < ApplicationController
  skip_before_action :verify_authenticity_token

  def create
    InboundWebhook.create(body: payload)
    head :ok
  end

  private

  def payload
    @payload ||= request.body.read
  end
end
```

Let's create a database record

```
curl -X POST http://localhost:3000/webhooks/movies
     -H 'Content-Type: application/json'
     -d '{"title":"Fellowship of the Ring"}'
```

You should now have an InboundWebhook record in your database.

```
rails c
```

```
InboundWebhook.count
```

Step 3: Processing Webhooks in the Background

```
rails generate job Webhooks::MoviesJob
```

```
rails generate job Webhooks::StripeJob
```

A job to process Movies webhooks

```ruby
# app/jobs/webhooks/movies_job.rb

class Webhooks::MoviesJob < ApplicationJob
  queue_as :default

  def perform(inbound_webhook)
    webhook_payload = JSON.parse(inbound_webhook.body, symbolize_names: true)

    # do whatever you'd like here with your webhook payload
    # call another service, update a record, etc.

    inbound_webhook.update!(status: :processed)
  end
end
```

A job to process our Stripe webhooks

```ruby
# app/jobs/webhooks/stripe_job.rb

class Webhooks::StripeJob < ApplicationJob
  queue_as :default

  def perform(inbound_webhook)
    json = JSON.parse(inbound_webhook.body, symbolize_names: true)
    event = Stripe::Event.construct_from(json)
    case event.type
    when 'customer.updated'
      # Find customer and save changes
      inbound_webhook.update!(status: :processed)
    else
      inbound_webhook.update!(status: :skipped)
    end
  end
end
```

Now we need to trigger our jobs

Let's update our controllers

Our Webhooks::MoviesController

```ruby
# app/controllers/webhooks/movies_controller.rb

def create
  # Save webhook to database
  record = InboundWebhook.create!(body: payload)

  # Queue database record for processing
  Webhooks::MoviesJob.perform_later(record)

  head :ok
end
```

Our Webhooks::StripeController

```ruby
# app/controllers/webhooks/stripe_controller.rb

def create
  # Save webhook to database
  record = InboundWebhook.create!(body: payload)

  # Queue database record for processing
  Webhooks::StripeJob.perform_later(record)

  head :ok
end
```

Step 4: Verifying Our Webhook Payloads

- Verify that the webhook is real and from the provider

Let's add a `verify_event` method to our BaseController

```ruby
# app/controllers/webhooks/base_controller.rb

class Webhooks::BaseController < ApplicationController
  skip_before_action :verify_authenticity_token

  before_action :verify_event

  def create
    InboundWebhook.create(body: payload)
    head :ok
  end

  private

  def verify_event
    head :bad_request
  end

  def payload
    @payload ||= request.body.read
  end
end
```

Now we can implement `verify_event` specifically for each provider

Movies Webhook Verification

```ruby
# app/controllers/webhooks/movies_controller.rb
# ... rest of controller

private

def verify_event
  head :bad_request if params[:fail_verification]
end
```

```
curl -X POST http://localhost:3000/webhooks/movies?fail_verification=1
     -H 'Content-Type: application/json'
     -d '{"title":"Hackers"}'
```

Stripe Webhook Verification

```ruby
# app/controllers/webhooks/stripe_controller.rb
# ... rest of controller

private

def verify_event
  signature = request.headers['Stripe-Signature']
  secret = Rails.application.credentials.dig(:stripe, :webhook_signing_secret)

  ::Stripe::Webhook::Signature.verify_header(
    payload,
    signature,
    secret.to_s,
    tolerance: Stripe::Webhook::DEFAULT_TOLERANCE,
  )
rescue ::Stripe::SignatureVerificationError
  head :bad_request
end
```

Step 5: Adding Some Tests

```
mkdir test/fixtures/webhooks
```

```
touch test/fixtures/webhooks/movie.json
```

```
{
  "title": "Dungeons & Dragons: Honor Among Thieves",
  "release_date": "2023-03-31"
}
```

Our First Webhook Test

```ruby
# spec/requests/webhooks/movies_controller_spec.rb
require 'test_helper'

class Webhooks::MoviesControllerTest < ActionDispatch::IntegrationTest
  def setup
    # Load the webhook data from the JSON file
    file_path = Rails.root.join('test', 'fixtures', 'webhooks', 'movie.json')
    @webhook = JSON.parse(File.read(file_path))
  end

  test 'should consume webhook' do
    # Send the POST request to the create action with the prepared data
    post webhooks_movies_url, params: @webhook

    # Check if the response status is 200 OK
    assert_response :ok

    # You can create other test files for each job or service that is called
    # For example, check if a record was created/updated in the database
  end
end
```

Similarity to Action Mailbox

Security

- Verification methods
  - TLS, Oauth, Asymetric keys, HMAC
- Replay Attacks
- Dataless notifications

Scaling

- Background processing
- AWS Eventbridge

Refactoring As Your Codebase Grows

- Middleware
- Webhook Handler

Common Gotchas

- Provider could fail to send a webhook
- Provider might not retry sending webhooks
- Your application needs to be up to receive a webhook
- Webhooks may arrive out of order
- Data in webhook payload may be stale
- Webhooks may be duplicated

Tools

- Webhooks.fyi
- ngrok and other localtunnels
- Hookrelay

🪝 Questions?

Thank You!

 @colinloretz and @excid3