Bending Time With Crystal

RubyConf 2022

Paul Hoffer

Who Am I?

- Full time w/ Rails
- Focus on performance & architecture
- Crystal ⇔ Ruby experience
  - Native extensions in Crystal
- Just like to experiment with big problems

Paul Hoffer

phoffer

LinkedIn

The RealReal

What is The RealReal?

- We are an ecommerce platform
- Luxury Consignment
  - Designer bags, watches, clothing, many more
  - Everything is authenticated
- Architecture
  - Rails monolith
  - Numerous Phoenix (Elixir) front end apps
  - Slowly extracting more things from Rails to smaller service apps

Introduction to Crystal

- "A Language for Humans and Computers"
- Ruby's efficiency for writing code
  - Ruby-like syntax
- C's efficiency for running code
  - Compiled with LLVM

Sample Crystal

```ruby
(1..9).each do |num|
  if num.even?
    puts "#{num} is even"
  else
    puts "#{num} is odd"
  end
end

3.times do
  num = rand(20)
  if num.modulo(4).zero?
    puts "#{num} is divisible by 4"
  elsif num < 10
    puts "#{num} has 1 digit"
  else
    puts "#{num} has multiple digits"
  end
end
```

Output (same in Ruby too!)

```
1 is odd
2 is even
3 is odd
4 is even
5 is odd
6 is even
7 is odd
8 is even
9 is odd
12 is divisible by 4
3 has 1 digit
14 has multiple digits
```

Crystal Ecosystem

- Shards ⇔ RubyGems + Bundler
- https://github.com/veelenga/awesome-crystal
- Wide range of shards
  - Web frameworks similar to Rails and Sinatra
  - Database tools similar to ActiveRecord, Ecto (Elixir)
  - Sidekiq.cr, mailers, etc
- Sometimes, shards can be ported from an existing RubyGem

What does this mean for us?

- Crystal code can be easy to understand and familiar to write
- There is likely an existing library for our specific use case
- Contributing can be relatively straightforward
- Crystal is a great tool for Rubyists!

Current Problem

Sitemap generation for TheRealReal

- Generates links for all our shopping pages for the following models
  - Sales
  - Categories
  - Designers
  - Promotions
  - Products
- Business related links
  - About, Press, Privacy
  - Shipping, Returns
  - Generic product landing pages, designers, categories

What makes it so difficult?

- 18 million links
  - Almost entirely products currently for sale
- 6 hours time duration to generate
- Updated daily, runs overnight
- Very memory hungry
  - Links are held in memory until final generation
  - Full ActiveRecord model objects are loaded
- Shopping front end is delivered by separate app
  - Rails code exists solely to support sitemap

Is sitemap generation really that complex?

NO!

Current Sitemap Generation

```
SitemapGenerator::Sitemap.create do
  add "/about",          changefreq: 'weekly'
  add "/team",           changefreq: 'weekly'

  fetch_products.find_each do |product|
    add product_path(product.permalink), lastmod: product.updated_at
  end

  FlashSale.active.find_each do |flash_sale|
    add flash_sale_path(flash_sale.permalink), lastmod: flash_sale.updated_at
  end
end
```

Could we use Crystal for this?

Why would we want to?

- Make it fast
  - Flexibility to run on a different schedule
- Reduce memory usage
  - Lower server instance requirements
- Improve long term sustainability
  - Recurring tasks that continue to grow will eventually become problematic
- Remove code that isn't used anywhere else in the Rails app

How could we use Crystal?

- What's the scope and what tools are necessary?
  - Database modeling
  - Path helpers for routing
  - Sitemap creation
- Is there tooling to make this easier?
  - Sitemap - https://github.com/jwoertink/sitemapper
  - Database - https://github.com/imdrasil/jennifer.cr
- Other things
  - 5 path helpers now unused by Rails except for sitemap

Biggest question:

How difficult would it be to port the sitemap generation logic?

Ruby Sitemap Generation

```ruby
SitemapGenerator::Sitemap.create do
  add "/about",           changefreq: 'weekly'
  add "/team",            changefreq: 'weekly'

  fetch_products.find_each do |product|
    add product_path(product.permalink), lastmod: product.updated_at
  end

  FlashSale.active.find_each do |flash_sale|
    add flash_sale_path(flash_sale.permalink), lastmod: flash_sale.updated_at
  end
end
```

Crystal Sitemap Generation

```
Sitemapper.create do |builder|
  builder.add "/about",          changefreq: 'weekly'
  builder.add "/team",           changefreq: 'weekly'

  fetch_products.find_each do |product|
    builder.add product_path(product.permalink), lastmod: product.updated_at
  end

  FlashSale.active.find_each do |flash_sale|
    builder.add flash_sale_path(flash_sale.permalink), lastmod: flash_sale.updated_at
  end
end
```

Ok, let's build it!

First Step – Database Modeling

- Jennifer.cr
- Similar query API to ActiveRecord
- Includes scopes and associations
- Goal: minimize code changes

First Step – Database Modeling

```
class LandingPage < Jennifer::Model::Base
  with_timestamps

  mapping({
    id:          Primary32,
    designer_id: {type: Int32, null: false},
    taxon_id:    {type: Int32, null: true},
  }, false)

  belongs_to :designer, Designer
  belongs_to :taxon,    Taxon

  scope :has_designer { where { _designer_id != nil } }
end
```

Database Interaction

```
LandingPage.has_designer.eager_load(:designer, :taxon)
Product.available
Sale.active.find_each do |sale|
  { id: sale.id, permalink: sale.permalink }
end
```

Next step – Generation code

- Sitemapper
- Similar API to SitemapGenerator RubyGem
- Same configuration options
- Same functionality
  - Compression
  - Upload to S3
  - Notify search engines
- Goal: minimize code changes

Crystal Sitemap Generation

```
Sitemapper.create do |builder|
  builder.add "/about",          changefreq: 'weekly'
  builder.add "/team",           changefreq: 'weekly'

  fetch_products.find_each do |product|
    builder.add product_path(product.permalink), lastmod: product.updated_at
  end

  FlashSale.active.find_each do |flash_sale|
    builder.add flash_sale_path(flash_sale.permalink), lastmod: flash_sale.updated_at
  end
end
```

Crystal Sitemap Generation

```
Sitemapper.create do |builder|
  builder.add "/about",          changefreq: 'weekly'
  builder.add "/team",           changefreq: 'weekly'

  fetch_products.find_each do |product|
    builder.add product_path(product.permalink), lastmod: product.updated_at
  end

  FlashSale.active.find_each do |flash_sale|
    builder.add flash_sale_path(flash_sale.permalink), lastmod: flash_sale.updated_at
  end
end
```

Crystal Sitemap Generation

```
Sitemapper.create do |builder|
  builder.add "/about",          changefreq: 'weekly'
  builder.add "/team",           changefreq: 'weekly'

  fetch_products.find_each do |product|
    builder.add product_path(product.permalink), lastmod: product.updated_at
  end

  FlashSale.active.find_each do |flash_sale|
    builder.add flash_sale_path(flash_sale.permalink), lastmod: flash_sale.updated_at
  end
end
```

Path and data helpers

```
def product_path(permalink)
  "/products/" + permalink
end

def flash_sale_path(permalink)
  "/flash_sales/" + permalink
end

def fetch_products
  Product.available.not_gift_card
end
```

Crystal Sitemap Generation

```
Sitemapper.create do |builder|
  builder.add "/about",          changefreq: 'weekly'
  builder.add "/team",           changefreq: 'weekly'

  fetch_products.find_each do |product|
    builder.add product_path(product.permalink), lastmod: product.updated_at
  end

  FlashSale.active.find_each do |flash_sale|
    builder.add flash_sale_path(flash_sale.permalink), lastmod: flash_sale.updated_at
  end
end
```

Does it work?!

Does it work?!

Yes! (and no)

So close!

- It's incredibly fast!
  - 15 minutes on dev machine, down from 6 hours
- Still suffering a memory leak
  - Crystal library also accumulates links until the end
- Maybe we can fix this?
  - Write out files as we accumulate links
  - Reset links when writing out

Contributing to Crystal libraries

streaming files functionality #33

Merged jwoertink merged 3 commits into `jwoertink:master` from `phoffer:streaming-writes` on Aug 13

Conversation 10 | Commits 3 | Checks 0 | Files changed 3

phoffer commented on Aug 12 • edited — Contributor

Ability to write out files as they are generated. For large datasets, can have a huge memory reduction (and keep it stable).

This is lacking new specs, but all existing specs still pass locally. Not yet tested with AWS storage.

Implements #32

P.S. It's been a long time since I've touched Crystal, feel free for any feedback at all

❤️ 1

Final Results

```
Generation Time : 892.4
Products        : 18396336
Flash Sales     : 2
Taxons          : 2073
Landing Pages   : 690
Evergreen Sales : 713
Peak RAM usage  : 1.2GB
```

Completed Prototype

What all went into this?

- 1 day to get a functional prototype
- 1 day to fix the memory leak and optimize it for Crystal
- 1 PR (Sitemapper)
- Less time than preparing for this talk!

```
require "sitemapper"
require "./models"

def product_path(permalink)
  "/products/" + permalink
end

def flash_sale_path(permalink)
  "/flash_sales/" + permalink
end

def taxon_landing_page_path(permalink)
  "/shop/" + permalink
end

def sale_path(permalink)
  "/sales/" + permalink
end

def landing_page_path(designer : Designer, taxon)
  path = if taxon
    "#{designer.slug}/#{taxon.permalink}"
  else
    designer.slug
  end
  "/designers/" + path
end

def fetch_products
  Spree::Product.all.available.not_deleted.representative.not_gift_card
end

def uri_decode(path)
  URI.decode_www_form(path)
end

start_time = Time.utc
Sitemapper.configure do |c|
  c.use_index    = true
  c.host         = "http://localhost"
  c.sitemap_host = "https://assets-staging.therealreal.com/"
  c.max_urls     = 50_000
  c.compress     = false
  c.storage      = Sitemapper::LocalStorage
end

Sitemapper.build do |builder|
  builder.add "/about",          changefreq: "weekly"
  builder.add "/team",           changefreq: "weekly"
  builder.add "/press",          changefreq: "weekly"
  builder.add "/consignments",   changefreq: "weekly"
  builder.add "/shipping",       changefreq: "weekly"
  builder.add "/designers",      changefreq: "weekly"
  builder.add "/faq",            changefreq: "weekly"
  builder.add "/[illegible]",    changefreq: "weekly"
  builder.add "/retail",         changefreq: "weekly"
  builder.add "/terms",          changefreq: "weekly"
  builder.add "/privacy",        changefreq: "weekly"
  builder.add "/mobile",         changefreq: "weekly"
  builder.add "/authenticity",   [illegible]: "weekly"

  fetch_products.find_each do |product|
    builder.add uri_decode(product_path(product.permalink)), lastmod: product.updated_at, changefreq: "daily"
  end

  [illegible].in_display_state.active.each do |flash_sale|
    builder.add flash_sale_path(flash_sale.permalink), lastmod: flash_sale.updated_at, changefreq: "daily"
  end

  Spree::Taxon.all.each do |taxon|
    builder.add uri_decode(taxon_landing_page_path(taxon.permalink)), lastmod: taxon.updated_at, changefreq: "weekly"
  end

  [illegible].has_designer.eager_load((:designer, :taxon)).find_each do |page|
    builder.add uri_decode(landing_page_path(page.designer, page.taxon)), changefreq: "daily"
  end

  EvergreenSale.active.find_each do |evergreen_sale|
    builder.add sale_path(evergreen_sale.permalink), lastmod: evergreen_sale.updated_at, changefreq: "daily"
  end
end

puts <<-SUMMARY
Generation Time : #{(Time.utc - start_time).to_f.round(1)}
Products        : #{fetch_products.count}
Flash Sales     : #[illegible].in_display_state.active.count}
Taxons          : #{Spree::Taxon.all.count}
Landing Pages   : #[illegible].has_designer.count}
Evergreen Sales : #[illegible].active.count}
SUMMARY
```

```
require "jennifer"
require "jennifer/adapter/mysql"
require "./jennifer_patch"
require "yaml"
require "ecr"

APP_ENV = ENV["APP_ENV"]? || "development"
config = YAML.parse ECR.render("./config/database.yml")

Jennifer::Config.configure do |conf|
  if ENV.has_key?("DATABASE_PRIMARY_URL")
    conf.from_uri(ENV["DATABASE_PRIMARY_URL"].gsub("mysql2", "mysql"))
  else
    conf.from_yaml(config[APP_ENV])
  end
  conf.logger.level = Log::Severity::Error
end

class LandingPage < Jennifer::Model::Base
  with_timestamps

  mapping({
    id: [illegible],
    designer_id: {type: Int32, null: false },
    taxon_id: {type: Int32, null: true },
    updated_at: Time,
  }, false)

  belongs_to :designer, Designer
  belongs_to :taxon, Spree::Taxon, foreign: "taxon_id"

  scope :has_designer { where { _designer_id != nil } }
end

class Designer < Jennifer::Model::Base
  with_timestamps

  mapping({
    id: [illegible],
    slug: String,
    updated_at: Time
  }, false)

  has_many :landing_pages, [illegible]
end

class Sale < Jennifer::Model::Base
  with_timestamps

  mapping({
    id: [illegible],
    permalink: String,
    type: [illegible],
    updated_at: [illegible]
    [illegible]

  scope :active { where { s("draft") == false } }
  scope :in_display_state { where { s("ends_at") > Time.utc } }
end
class [illegible] < Sale; end
class [illegible] < Sale; end

module Spree
  class Taxon < Jennifer::Model::Base
    with_timestamps

    mapping({
      id: [illegible],
      permalink: String,
      updated_at: Time,
    }, false)

    has_many :landing_pages, [illegible]
  end

  class Product < Jennifer::Model::Base
    with_timestamps

    mapping({
      id: [illegible],
      permalink: String,
      available_at: Time,
      is_gift_card: Bool,
      deleted_at: {type: Time, null: true},
      updated_at: Time,
    }, false)

    scope :visible { where { and(_available_at != nil, _deleted_at == nil) } }
    scope :available { where { _available_at != nil } }
    scope :not_deleted { where { _deleted_at == nil } }
    scope :not_gift_card { where { _is_gift_card == false } }
    scope :representative { where { sql("representative_id IS NULL OR representative_id = spree_products.id") } }
    scope :launching_after { |launch_at| where { _available_at > launch_at } }
  end
end
```

185 lines of code!

Bending Time With Crystal

Recapping the creative process

- Examining what the problem is
  - Realizing it was very loosely coupled to Rails
  - It could potentially be extracted to a separate service
- How to solve
  - We're familiar with Ruby and Crystal
  - Similar Crystal libraries allow us to test without wasting too much time
  - Utilizing Ruby knowledge to make contributions to Crystal libraries
- A general sense of fun while chasing down problems

Thank you 💚

RubyConf!

All of you!

The RealReal!

(p.s. we're hiring!)

Paul Hoffer

phoffer

LinkedIn

The RealReal

Resources

- https://crystal-lang.org
- https://carc.in
- https://gitter.im/crystal-lang/crystal
- https://www.crystalforrubyists.com
- https://github.com/crystal-lang/crystal/wiki/Crystal-Shards-for-Ruby-Gems

Questions?

Paul Hoffer

phoffer

LinkedIn

The RealReal