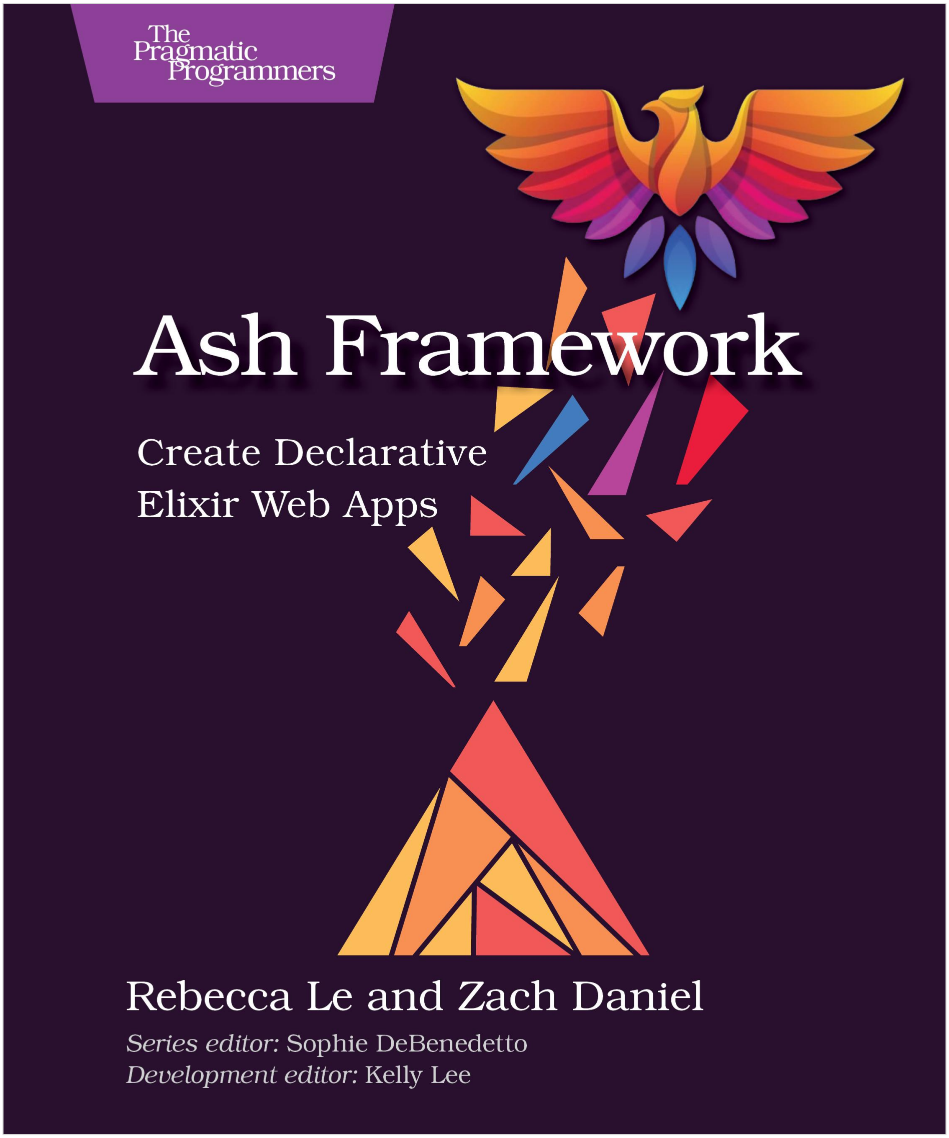


The Pragmatic Programmers

Ash Framework

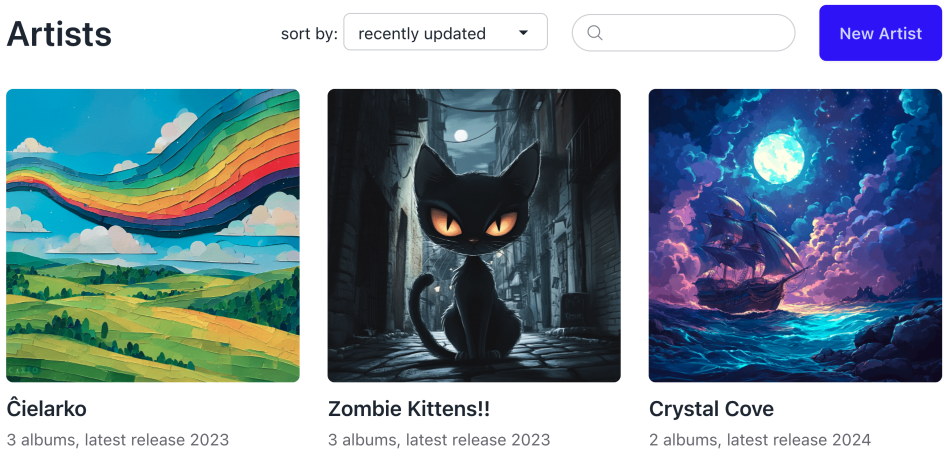
Create Declarative Elixir Web Apps

Rebecca Le and Zach Daniel

Series editor: Sophie DeBenedetto
Development editor: Kelly Lee

Using aggregates like any other attribute

It would be amazing if the artist catalog looked like a beautiful display, of album artwork and artist information.



Ĉielarko
3 albums, latest release 2023

Zombie Kittens!!
3 albums, latest release 2023

Crystal Cove
2 albums, latest release 2024

In Tunez.Artists.IndexLive, the cover image display is handled by a call to the cover_image function component within artist_card:

03/lib/tunez_web/live/artists/index_live.ex

```
<div id={"artist-#{@artist.id}"} data-role="artist-card" class="relative mb-2">
  <.link navigate={~p"/artists/#{@artist}"}>
    <.cover_image />
  </.link>
</div>
```

Because we can use and reference aggregate attributes like any other attributes on a resource, we can add an image argument to the cover_image function component, to replace the default placeholder image with our cover_image_url calculation:

03/lib/tunez_web/live/artists/index_live.ex

```
<div id={"artist-#{@artist.id}"} data-role="artist-card" class="relative mb-2">
  <.link navigate={~p"/artists/#{@artist}"}>
➤   <.cover_image image={@artist.cover_image_url} />
  </.link>
</div>
```

Refreshing the artist catalog after making the change might not be what you expect — why aren't the covers displaying? Because we aren't loading them! Remember that we need to specifically load calculations/aggregates if we want to use them, they won't be generated automatically.

Because we'll want all of our new aggregates loaded every time we run the search action on the Artist resource, and it's not a dynamic list in any way, we can add a preparation[18] to that action to load our data.

03/lib/tunez/music/artist.ex

```
read :search do
  # ...

  prepare build(load: [:album_count, :latest_album_year_released,
    :cover_image_url])
end
```

Reloading the artist catalog will now populate the data for all of the aggregates we listed, and look at the awesome artwork appear! Special thanks to Midjourney, for bringing our imagination to life!

For the album count and latest album year released fields, we can add those details to the end of the artist_card function, using the previously-unused artist_card_album_info component defined right below it:

03/lib/tunez_web/live/artists/index_live.ex

```
def artist_card(assigns) do
  ~H"""
  # ...
  <.artist_card_album_info artist={@artist} />
  """
end
```

And behold! The artist catalog is now in its full glory!

Earlier in the chapter, we looked at sorting artists in the catalog, via three different attributes - name, inserted_at and updated_at. We've explicitly said a few times now, that calculations and aggregates can be treated like any other attribute — does that mean we might be able to sort on them too???

You bet you can!

Sorting based on aggregate data

Around this point is where Ash really starts to shine, and you might start feeling a bit of a tingle with the power at your fingertips. Hold that thought, because it's going to get even better. Let's add some new sort options for our aggregate attributes, to our list of available sort options in Tunez.Music.IndexLive:

03/lib/tunez_web/live/artists/index_live.ex

```
def sort_options do
  [
```

18. https://hexdocs.pm/ash/Ash.Resource.Preparation.Builtins.html#build/1

```
      {"-updated_at", "recently updated"},
      {"-inserted_at", "recently added"},
      {"name", "name"},
➤     {"-album_count", "number of albums"},
➤     {"--latest_album_year_released", "latest album release"}
    ]
  end
```

We want artists with the most albums and with the most recent albums listed first, so we'll sort them descending by prefixing the attribute name with a -. Using -- is a bit special - it'll put any nil values (if an artist hasn't released any albums!) at the end of the list.

To allow the aggregates to be sorted on, we do need to mark them as public? true, like we did with our initial set of sortable attributes in Using sort_input for succinct yet expressive sorting, on page ?:

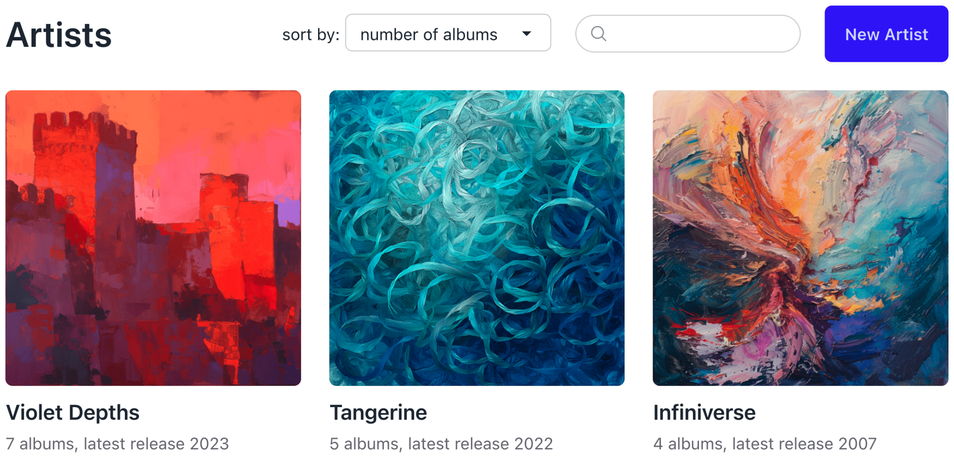
03/lib/tunez/music/artist.ex

```
  aggregates do
    count :album_count, :albums do
➤     public? true
    end

    first :latest_album_year_released, :albums, :year_released do
➤     public? true
    end

    # ...
  end
```

And then we'll be able to sort in our artist catalog, to see which artists have the most albums, or have released albums most recently:



Violet Depths
7 albums, latest release 2023

Tangerine
5 albums, latest release 2022

Infiniverse
4 albums, latest release 2007

This is all *amazing*! We've built some really excellent functionality over the course of this chapter, to let users search, sort, and paginate through data.

And in the next one, we'll see how we can use the power of Ash to build some neat APIs for Tunez, using our existing resources and actions. Reduce, re-use and recycle code!