PostgREST 12

2023-12-13 • 4 minute read

PostgREST 12 is out. In this post, we'll focus on a few of the major features. For the complete list, check out the release on GitHub.

Performance: JWT Caching

Until now, PostgREST has validated JWTs on every request. As of PostgREST 12, the JWT is cached on the first request using the `exp` claim to set the cache entry's lifetime.

Why is that a big deal? Well, it turns out decoding JWTs is expensive. Very expensive.

```
## before
$ curl 'localhost:3000/authors_only' -H "Authorization: Bea
HTTP/1.1 200 OK
Server-Timin

## after, wi
$ curl 'loca
HTTP/1.1 200 OK
```

```
9  Server-Timing: jwt;dur=14.1
```

The JWT cache shaves over 130ms off the server side timing. For projects with a high volume of API calls, upgrading to PostgREST 12 gives you faster responses, higher throughput, and lower resource consumption.

Server Timing Header

Did you notice the `Server-Timing` header in the last example? That's new too and it does more than measure JWT decoding duration.

Here's a complete reference to what you can extract from your responses:

```
1 Server-Timing:
2     jwt;dur=14.9,
3     parse;dur=71.1,
4     plan;dur=109.0,
5     transaction;dur=353.2,
6     response;dur=4.4
```

Where the information from each phase is internally timed by PostgREST for better visibility into server side performance.

Aggregate Functions

Support for aggregate functions has been much requested feature that went through multiple iterations of design and review.

Currently, PostgREST supports `avg`, `count`, `max`, `min`, `sum`. Here's a minimal example using `count`:

```
1 $ curl 'http://postgrest/blog_post?select=id.count()'
2
3 [
4   {
5     "count":
6   }
7 ]
```

We can also add a “group by” simply by adding another element to the select clause.

```
$ curl 'http://postgrest/blog_post?select=title,id.count()

[
  {
    "title": "Supabase Blog",
    "count": 40
  },
  {
    "title": "Contributors Blog",
    "count": 11
  },
  ...
```

This example only scratches the surface. Aggregates are fully-compatible with resource embedding which yields an extremely versatile interface. We'll explore this feature more in a deep-dive coming soon.

Media Type Handlers

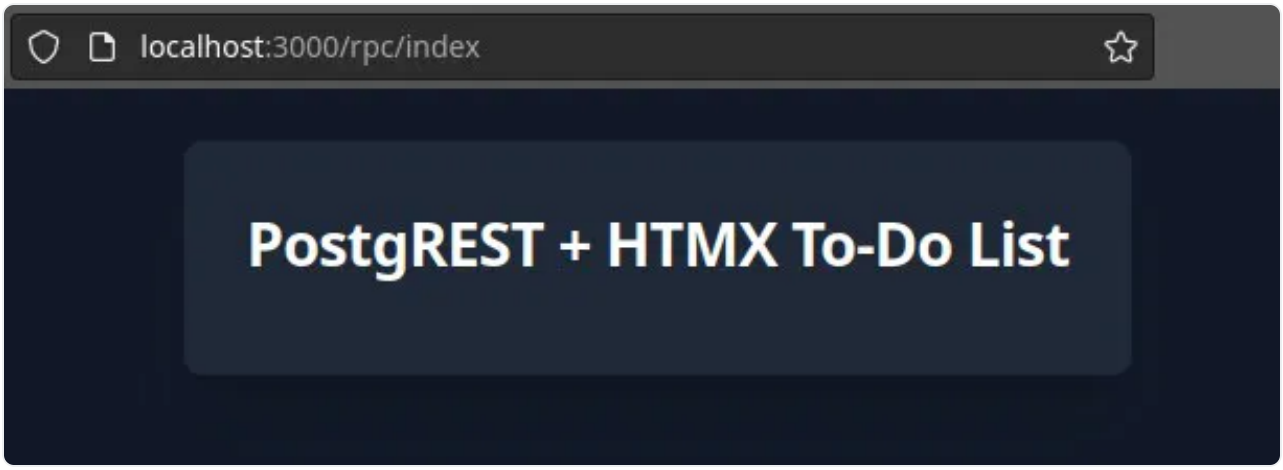
PostgREST now gives you the flexibility to handle your custom media types and override the built-in ones. Among other things, that enables serving HTML, javascript, or whatever you can think of, straight from your database.

```
create domain "text/html" as text;

create or replace function api.index()
returns "text/html"
language sql
as $$
  select $html$
    <!DOCTYPE html>
    <html>
    <head>
      <meta charset="utf-8">
      <meta name="viewport" content="width=device-width, i
      <title>PostgREST + HTMX To-Do List</title>
      <!-- Tailwind for CSS styling -->
      <link href="https://unpkg.com/tailwindcss@2.2.19/dist
    </head>
    <body cl
      <div c
        <div
          <h
        </div
```

```
22        </div>
23      </body>
24      </html>
25    $html$;
26  $$;
```

With PostgREST running locally we can then navigate to localhost:3000/rpc/index to see



We're still working through the full implications of this feature, but we're very excited internally about the possibilities it unlocks! Similar to aggregate functions, there's a dedicated post for this feature on the way.

Availability

For self-hosting, check out the PostgREST release on GitHub.

The latest version will be rolled out across all projects on the managed platform soon. Keep an eye out for notifications inside Supabase Studio.

More Launch Week X

- Day 1 - Supabase Studio update: AI Assistant and User Impersonation
- Day 2 - Edge Functions: Node and native npm compatibility
- Day 3 - Supabase Branching
- pg_graphql: Pos
- Postgres Langua

The Supabase Album

Launch Week X Hackathon

Share this article

Last post

Supavisor 1.0: a scalable connection pooler for Postgres
13 December 2023

Next post

Edge Functions: Node and native npm compatibility
12 December 2023

Related articles

Supabase Wrappers v0.2: Query Pushdown & Remote Subqueries
Supabase Branching
Supavisor 1.0: a scalable connection pooler for Postgres
Edge Functions: Node and native npm compatibility
pg_graphql: Postgres functions now supported

View all posts

Build in a weekend, scale to millions

Start your project