GPT-4 for API design research

I came up with a really useful prompt for GPT-4 today. I was considering options for refactoring how Datasette's core view functions work, and was contemplating alternative ways to dispatch to different functions based on a combination of the URL path and the HTTP verb.

I decided it would be useful to see how other frameworks solved that, and came up with the following prompt:

> Show me code examples of different web frameworks in Python and JavaScript and Go illustrating how HTTP routing works - in particular the problem of mapping an incoming HTTP request to some code based on both the URL path and the HTTP verb
>
> Don't show installation instructions, just get straight to the code snippets

(I added that second line after it first started spitting out installation instructions for each framework, when I just wanted to see some illustrative code examples.)

I got really great results!

It started by kicking out code examples for Flask (Python), Express (JavaScript/Node.js) and Gin (Go).

Then I prompted:

> Three more examples for the same languages but showing different frameworks

It gave me Django (Python), Koa (JavaScript/Node.js) and Echo (Go).

Finally, I added:

> Some Rust examples too

And it gave me examples for Actix-web and Rocket, both in Rust.

I've shared the full transcript here: https://gist.github.com/simonw/fa2379b97420404a81b0fcdb4db79657

As someone who spends a whole lot of time on API design, this feels like a *really* useful trick - researching how multiple frameworks across multiple other languages solve a problem is incredibly useful for coming up with a coherent new API design.

The examples it gave me

Here's just the code from the examples I found most useful:

Flask:

```
from flask import Flask

app = Flask(__name__)
```

```python
@app.route('/example', methods=['GET'])
def get_example():
    return "This is a GET request."

@app.route('/example', methods=['POST'])
def post_example():
    return "This is a POST request."

if __name__ == '__main__':
    app.run()
```

Express:

```javascript
const express = require('express');
const app = express();

app.get('/example', (req, res) => {
  res.send('This is a GET request.');
});

app.post('/example', (req, res) => {
  res.send('This is a POST request.');
});

app.listen(3000, () => {
  console.log('Server is running on port 3000');
});
```

Gin:

```go
package main

import (
        "github.com/gin-gonic/gin"
)

func main() {
        r := gin.Default()

        r.GET("/example", func(c *gin.Context) {
                c.String(200, "This is a GET request.")
        })

        r.POST("/example", func(c *gin.Context) {
                c.String(200, "This is a POST request.")
        })

        r.Run()
}
```

Actix-web:

```rust
use actix_web::{web, App, HttpResponse, HttpServer, Responder};

async fn get_example() -> impl Responder {
    HttpResponse::Ok().body("This is a GET request.")
}
```

```
async fn post_example() -> impl Responder {
    HttpResponse::Ok().body("This is a POST request.")
}

#[actix_web::main]
async fn main() -> std::io::Result<()> {
    HttpServer::new(|| {
        App::new()
            .route("/example", web::get().to(get_example))
            .route("/example", web::post().to(post_example))
    })
    .bind("127.0.0.1:8080")?
    .run()
    .await
}
```

Koa:

```
const Koa = require('koa');
const Router = require('koa-router');

const app = new Koa();
const router = new Router();

router.get('/example', (ctx) => {
  ctx.body = 'This is a GET request.';
});

router.post('/example', (ctx) => {
  ctx.body = 'This is a POST request.';
});

app.use(router.routes());
app.use(router.allowedMethods());

app.listen(3000, () => {
  console.log('Server is running on port 3000');
});
```

Django:

```
from django.http import HttpResponse
from django.views import View

class ExampleView(View):
    def get(self, request):
        return HttpResponse("This is a GET request.")

    def post(self, request):
        return HttpResponse("This is a POST request.")

from django.urls import path
from .views import ExampleView

urlpatterns = [
    path('example/', ExampleView.as_view(), name='example'),
]
```

Rocket:

```
#![feature(proc_macro_hygiene, decl_macro)]

#[macro_use] extern crate rocket;

use rocket::response::content;

#[get("/example")]
fn get_example() -> content::Plain<&'static str> {
    content::Plain("This is a GET request.")
}

#[post("/example")]
fn post_example() -> content::Plain<&'static str> {
    content::Plain("This is a POST request.")
}

#[launch]
fn rocket() -> _ {
    rocket::build().mount("/", routes![get_example, post_example])
}
```

And from a tip from Jim Flanagan, here's what it gave me for Elixir/Phoenix:

```
defmodule ExampleAppWeb.Router do
  use ExampleAppWeb, :router

  pipeline :api do
    plug :accepts, ["json"]
  end

  scope "/api", ExampleAppWeb do
    pipe_through :api

    get "/example", ExampleController, :get_example
    post "/example", ExampleController, :post_example
  end
end
```

Related

python Simple load testing with Locust - 2022-10-22

deno Running Python code in a Pyodide sandbox via Deno - 2023-05-10

discord A Discord bot to expand issue links to a private GitHub repository - 2023-06-29

macos Running Docker on an M1 Mac - 2021-05-25

gpt3 Using ChatGPT Browse to name a Python package - 2023-06-18

Created 2023-04-06T12:17:57-07:00, updated 2023-04-06T12:41:20-07:00 · History · Edit