How to Perform a GeoIP Lookup in Flask

Posted on June 19, 2018

If you're building a website (or API) using Flask, it's often useful to know where your visitors are coming from: the US, the EU, someplace else? The process of locating a web user is typically referred to IP geolocation, and unfortunately, it isn't simple.

The reason it isn't easy to get IP geolocation data is that there is no standard mapping of IPs -> location data. Most companies get this data by purchasing it from GeoIP aggregators that piece together lots of different bits of information to build an accurate database of IP geolocation data.

GeoIP data is typically comprised of:

- WHOIS data
- Regional Internet Registries
- BGP feeds (from internet service providers)
- Latency information manually gathered

In short: it's basically impossible to get IP geolocation data without going through a GeoIP aggregator.

Today, I'll walk you through using the incredibly simple Flask-Simple-GeoIP developer library I created which makes performing GeoIP lookups a breeze in Flask.

Create a GeoIPify Lookup Account

The first thing you need to do is go create a free GeoIPify account: https://geoipify.whoisxmlapi.com/signup

GeoIPify is one of the biggest and cheapest GeoIP providers. You can use GeoIPify to perform 1,000 free GeoIP lookups per month, or you can pay them a flat fee of $27 for 100k lookups. If you need to do more than that, you can always check out the pricing (which is inexpensive): https://geoipify.whoisxmlapi.com/pricing

Once you've created your free account and logged in, you need to view the product page and copy your API key, you will need this later in order to make API requests to perform the GeoIP lookups.

Install Flask-Simple-GeoIP

Once you've created your GeoIPify account, you then need to install the Flask-Simple-GeoIP PyPI library. To do this, run the following command (I'm assuming you already have Flask setup and working):

```
$ pip install flask-simple-geoip
```

Plug Flask-Simple-GeoIP Into Your Flask App

Now that you've got the Flask-Simple-GeoIP library installed, all you need to do is hook it up into your Flask app. To do this, you'll need to import the library, initialize the extension, then test it out.

Here's a simple Flask app that only contains a single endpoint, `/test`, which returns a simple hello world response:

```
from flask import Flask
from flask_simple_geoip import SimpleGeoIP

app = Flask(__name__)
app.config["GEOIPIFY_API_KEY"] = "your-api-key-here"

simple_geoip = SimpleGeoIP(app)

@app.route("/test")
def test():
    return "hello, world!"
```

If you put this code into an `app.py` file, and run it, you should see a hello world response when you visit the `/test` endpoint in your browser.

The code above initializes the extension and prepares it for usage. Don't forget to substitute in the correct API key, however, or nothing will work.

NOTE: You can either set your API key via the app config (like in the example above) or by setting it as an environment variable of the same name. On Linux and Mac, you can set an environment variable by running the command `export GEOIPIFY_API_KEY=your-api-key-here` in the terminal before running your app.

To access the GeoIP data in your code, you can simply call the `get_geoip_data` the method:

```
from flask import Flask, jsonify
from flask_simple_geoip import SimpleGeoIP

app = Flask(__name__)
app.config["GEOIPIFY_API_KEY"] = "your-api-key-here"

simple_geoip = SimpleGeoIP(app)

@app.route("/test")
def test():
    geoip_data = simple_geoip.get_geoip_data()
    return jsonify(data=geoip_data)
```

If you run this new server and visit the `'/test` URL in your browser, you should see a new response that looks something like this:

```
{
  "ip": "8.8.8.8",
  "location": {
    "country": "US",
    "region": "California",
    "city": "Mountain View",
    "lat": 37.40599,
    "lng": -122.078514,
    "postalCode": "94043",
    "timezone": "-08:00"
  }
}
```

The `get_geoip_data` method returns a Python dictionary that contains all of the requester's geoip data in one nice place. Pretty neat, right? This dict contains everything you need to know about the requester's physical location.

How to Use GeoIP Data

Now that you've seen how simple it can be to get GeoIP lookup functionality working in your Flask apps, here are some ideas of ways you can use GeoIP data in your projects and services:

- Detect a user's country when they visit your site and provide a customized experience for them (change the language of the page, show certain ads, types of currency, etc.)
- Block users from certain locations from visiting your website. Let's say you are building a video streaming service like YouTube and you only have the rights to show certain videos to users in the US. In this case, having GeoIP data could help you detect a user's location so you could filter any non-US users.
- Reduce fraud and risk. If you notice a lot of malicious traffic coming from a specific country, temporarily blocking visitors from that location can be a quick way to avoid fraud and other issues.

Flask-Simple-GeoIP Wrap Up

Performing GeoIP lookups can be tricky, but Flask-Simple-GeoIP in conjunction with the GeoIPify service makes it simple and cheap. By using the new Flask-Simple-GeoIP library you can easily build and manage GeoIP lookups for even the largest enterprise sites.

To learn more, go check out the Flask-Simple-GeoIP library on GitHub where you can find all the docs and more in-depth information: https://github.com/whois-api-llc/flask-simple-geoip

If you have any questions, feel free to shoot me an email.