Practical Perl Tools
The Whether Man

DAVID N. BLANK-EDELMAN

David Blank-Edelman is the Technical Evangelist at David Blank-Edelman is the Technical Evangelist at Apcera (the comments/views here are David's alone and do not represent Apcera /Ericsson). He has spent close to 30 years in the system administration/DevOps/SRE field in large multiplatform environments including Brandeis University, Cambridge Technology Group, MIT Media Laboratory, and Northeastern University. He is the author of the O'Reilly Otter book *Automating System Administration with Perl* and is a frequent invited speaker/organizer for conferences in the field. David is honored to serve on the USENIX Board of Directors. He prefers to pronounce Evangelist with a hard 'g'.
dnblankedelman@gmail.com

There's a character in *The Phantom Tollbooth* by Norton Juster (one of my favorite books) called the Whether Man. He introduces himself like this:

"I'm the Whether Man, not the Weather Man, for after all it's more important to know whether there will be weather than what the weather will be."

In this column we're going to see if we can indeed bring weather to our programs. To do so, we're going to make use of the API provided by The Dark Sky Company, the people behind the popular app of the same name (and also the people who provided the API service forecast.io, renamed as of this week to the Dark Sky API). The API service they provide is a commercial one, so if you plan to build something that will hit their API more than a thousand times a day, you will need to pay. But for the experiments in this column, we should be pretty safe from the $0.0001 per forecast fee.

Two Flavors of API

The Dark Sky service offers two kinds of API, Forecast and the Time Machine (really). The first kind provides just a single forecast for next week's weather, the second will give us the opportunity to query for the conditions at an arbitrary time in the past or the future.

For both of these APIs, the actual method of querying and working with the data will be the same. Before we do anything, we'll need to sign up for a secret key. This key has to be sent along with every request we make. It's "secret" in that it shouldn't be distributed with your code. It's the thing that ties usage of the API back to your account (so you can pay for the service as needed). I'll be X'ing it out in all of the same code in this column.

Once we have a key, we can make an HTTPS request to the Dark Sky API endpoint. It is going to respond with a blob of (well-formed) JSON that we'll have to parse. Nothing special compared to our previous columns. Though there's an existing module for the previous name of the API (Forecast::IO), the tasks are so easy we'll just use more generic modules for what we need. Let's do some sample querying with both API types.

Whoops, one small complication that is worth mentioning: when you query either API, you will need to be specific about where you want to know about the weather. Whether there is weather is key but so is where is that weather. The Dark Sky API needs you to specify the latitude and longitude of the place before it can tell you the weather information for that place. The API will not geocode for you (i.e., translate an address to a latitude/longitude pair). There are a number of Web sites that will do small amounts of geocoding for you, but if you require more than a handful of translations, you will want to use a separate service for this part. We've talked about geocoding using Perl in this column before so some past columns in the archives are likely to be helpful. For this column I'm just going to use the coordinates of Boston (42.3600825, -71.0588801).

Give Me a Forecast

According to the Dark Sky docs, the format of the Forecast API is just:

```
https://api.darksky.net/forecast/[key]/[latitude],[longitude]
```

Let's query that URL and dump the data we get back. For fun I'm using the client that ships with Mojolicious and Data::TreeDumper which has purdy data structure dumps:

```
# note IO::Socket::SSL has to be installed for
# Mojo::UserAgent TLS support. It will fail silently
# if you don't
use Mojo::UserAgent;
use Data::TreeDumper;

my $API_KEY  = 'XXXXXXXXXXXXXXXXXXXX';
my $location = '42.3600825,-71.0588801';
my $endpoint = 'https://api.darksky.net/forecast';

my $ua = Mojo::UserAgent->new;

my $forecast =
  $ua->get( $endpoint . '/' . $API_KEY . '/' . $location )->
        res->json;

print "Powered by Dark Sky (https://darksky.net/poweredby/)\n";

print DumpTree( $forecast, 'forecast', DISPLAY_ADDRESS => 0 );
```

The Mojo::UserAgent call performs the GET (as long as we have IO::Socket::SSL installed; see the comment). We then take the result (->res) and parse the JSON we get back (->json). The end result is we get back a Perl data structure we can then access or manipulate to our heart's content.

Here's a dump of that data structure. This is an excerpt of the dump where I have removed all but the first instance of each kind of info (for example, just the first hour, not all of them). We get back lots of data (though if we didn't want all of this, there is an optional "exclude" flag we could pass, but where's the fun in that?):

```
Powered by Dark Sky (https://darksky.net/poweredby/)
forecast
|- currently
|  |- apparentTemperature = 72.96
|  |- cloudCover = 0.18
|  |- dewPoint = 56.87
|  |- humidity = 0.57
|  |- icon = clear-day
|  |- nearestStormBearing = 229
|  |- nearestStormDistance = 99
|  |- ozone = 261.5
|  |- precipIntensity = 0
|  |- precipProbability = 0
|  |- pressure = 1018.92
|  |- summary = Clear
|  |- temperature = 72.96
|  |- time = 1474582497
|  |- visibility = 10
|  |- windBearing = 116
|  `- windSpeed = 4.8
|- daily
|  |- data
|  |  |- 0
|  |  |  |- apparentTemperatureMax = 77.14
|  |  |  |- apparentTemperatureMaxTime = 1474560000
|  |  |  |- apparentTemperatureMin = 61.63
|  |  |  |- apparentTemperatureMinTime = 1474542000
|  |  |  |- cloudCover = 0.13
|  |  |  |- dewPoint = 56.15
|  |  |  |- humidity = 0.64
|  |  |  |- icon = partly-cloudy-night
|  |  |  |- moonPhase = 0.72
|  |  |  |- ozone = 262.42
|  |  |  |- precipIntensity = 0
|  |  |  |- precipIntensityMax = 0
|  |  |  |- precipProbability = 0
|  |  |  |- pressure = 1020.31
|  |  |  |- summary = Partly cloudy in the evening.
|  |  |  |- sunriseTime = 1474540400
|  |  |  |- sunsetTime = 1474584172
|  |  |  |- temperatureMax = 77.14
|  |  |  |- temperatureMaxTime = 1474560000
|  |  |  |- temperatureMin = 61.63
|  |  |  |- temperatureMinTime = 1474542000
|  |  |  |- time = 1474516800
|  |  |  |- visibility = 9.9
|  |  |  |- windBearing = 127
|  |  |  `- windSpeed = 1.27
|  |- icon = rain
|  `- summary = Light rain tomorrow and Saturday, with
temperatures
falling to 63°F on Monday.
|- flags
|  |- darksky-stations
|  |  `- 0 = KBOX
|  |- isd-stations
|  |  |- 0 = 725090-14739
|  |- lamp-stations
|  |  |- 0 = KASH
|  |- madis-stations
|  |  |- 0 = AV085
|  |- sources
|  |  |- 0 = darksky
|  `- units = us
|- hourly
```

```
|  |- data
|  |  |- 0
|  |  |  |- apparentTemperature = 73.52
|  |  |  |- cloudCover = 0.16
|  |  |  |- dewPoint = 56.82
|  |  |  |- humidity = 0.56
|  |  |  |- icon = clear-day
|  |  |  |- ozone = 261.61
|  |  |  |- precipIntensity = 0
|  |  |  |- precipProbability = 0
|  |  |  |- pressure = 1018.98
|  |  |  |- summary = Clear
|  |  |  |- temperature = 73.52
|  |  |  |- time = 1474581600
|  |  |  |- visibility = 10
|  |  |  |- windBearing = 109
|  |  |  `- windSpeed = 4.54
|  |- icon = rain
|  `- summary = Drizzle tomorrow evening.
|- latitude = 42.3600825
|- longitude = -71.0588801
|- minutely
|  |- data
|  |  |- 0
|  |  |  |- precipIntensity = 0
|  |  |  |- precipProbability = 0
|  |  |  `- time = 1474582440
|  |- icon = clear-day
|  `- summary = Clear for the hour.
|- offset = -4
`- timezone = America/New_York
```

The initial "Powered by" header is just my attempt to keep to the letter of the terms of service at https://darksky.net/dev/docs/terms#attribution.

There are a few important details that might not be obvious just looking at the dump above. First, the values we are getting back are in US-standard notation (e.g., Fahrenheit). This is the default, although we can send in a query parameter requesting the response be in metric units (or even automatically in the units of the place). There's a similar choice available for the human-readable summary fields above. Second, the time information we get back is in UNIX-standard time (i.e., seconds from the epoch). To translate that to local time, we could "scalar localtime $value."

The docs at https://darksky.net/dev/docs have the full details of the data we dumped. To access just parts of it, we could use simple code like this:

```
print "On " . scalar localtime(
            $forecast->{currently}->{time} ) . " it was:\n";
print $forecast->{currently}->{temperature} . " degrees F\n";
print( ( $forecast->{currently}->{humidity} * 100 )
       . "% relative humidity\n" );
```

Hop into My Time Machine

Let's move on to the second API, the "Time Machine" (cue the theremin). This API's request format is very similar to the last one (same endpoint and everything), it just adds the field you'd expect:

```
https://api.darksky.net/forecast/[key]/
[latitude],[longitude],[time]
```

The time field is either in the UNIX-standard time format we mentioned before or you can use this format:

```
[YYYY]-[MM]-[DD]T[HH]:[MM]:[SS][timezone]
```

(where time zone is optional).

Our code looks very similar:

```
use Mojo::UserAgent;
use Data::TreeDumper;

my $API_KEY  = 'XXXXXXXXX';
my $location = '42.3600825,-71.0588801';
my $endpoint = 'https://api.darksky.net/forecast';

my $exclude  = 'currently,minutely,hourly,alerts,flags';

# tomorrow
my $time = time + (24 * 60 * 60);

my $ua = Mojo::UserAgent->new;

my $forecast =
  $ua->get( $endpoint . '/'
                      . $API_KEY
                          . '/'
                          . $location
                          . ',' . $time
                          . '?exclude=' . $exclude )->res->json;

print "Powered by Dark Sky (https://darksky.net/poweredby/)\n";

print DumpTree( $forecast, 'forecast', DISPLAY_ADDRESS => 0 );
```

This code shows the daily info for tomorrow's forecast (because we've excluded the other portions we didn't want to see):

```
Powered by Dark Sky (https://darksky.net/poweredby/)
forecast
|- daily
|  `- data
|     `- 0
|        |- apparentTemperatureMax = 67.33
```

```
|       |- apparentTemperatureMaxTime = 1474747200
|       |- apparentTemperatureMin = 55.72
|       |- apparentTemperatureMinTime = 1474772400
|       |- cloudCover = 0.33
|       |- dewPoint = 46.6
|       |- humidity = 0.6
|       |- icon = partly-cloudy-night
|       |- moonPhase = 0.79
|       |- ozone = 302.59
|       |- precipIntensity = 0.0074
|       |- precipIntensityMax = 0.0965
|       |- precipIntensityMaxTime = 1474693200
|       |- precipProbability = 0.64
|       |- precipType = rain
|       |- pressure = 1017.92
|       |- summary = Mostly cloudy in the morning.
|       |- sunriseTime = 1474713327
|       |- sunsetTime = 1474756758
|       |- temperatureMax = 67.33
|       |- temperatureMaxTime = 1474747200
|       |- temperatureMin = 55.72
|       |- temperatureMinTime = 1474772400
|       |- time = 1474689600
|       |- visibility = 9.76
|       |- windBearing = 335
|       `- windSpeed = 8.82
|- latitude = 42.3600825
|- longitude = -71.0588801
|- offset = -4
`- timezone = America/New_York
```

To wrap this up, let's do something fun with the time machine API. There's a character in the Flintstones cartoon that always had a cloud following him around. If you ever wondered if the same thing happened to you, we could query the API to find out if the weather has been consistent for each of your birthdays. Here's some code that does this:

```
use Mojo::UserAgent;
use Text::Graph;

my $API_KEY  = 'XXXXXXXXXXXXX';
my $location = '42.3600825,-71.0588801';
my $endpoint = 'https://api.darksky.net/forecast';

my $exclude = 'currently,minutely,hourly,alerts,flags';

my $birth_year = 1970;

my $birth_date = "08-08";
my $current_year = 1900 + (localtime)[5];

my $collection;

my $ua = Mojo::UserAgent->new;

print "Powered by Dark Sky (https://darksky.net/poweredby/)\n";
for my $year ( $birth_year .. $current_year ) {
   my $forecast =
    $ua->get( $endpoint . '/'
       . $API_KEY . '/'
       . $location . ','
       . "$year-${birth_date}T12:01:01"
       . '?exclude='
       . $exclude )->res->json;

    print "$year:
          $forecast->{daily}->{data}->[0]->{summary}\n";

   $collection{ $forecast->{daily}->{data}->[0]->{icon} }++;

}

my $graph = Text::Graph->new('Bar');
print "\n",$graph->to_string( \%collection );
```

In case you are curious, that's not actually my birthday. And yes, this isn't quite accurate because I haven't been at those coordinates my entire life. But if you can suspend disbelief for a moment, check out the output:

```
Powered by Dark Sky (https://darksky.net/poweredby/)
1970: Partly cloudy throughout the day.
1971: Mostly cloudy throughout the day.
1972: Partly cloudy until evening.
1973: Mostly cloudy throughout the day.
1974: Mostly cloudy throughout the day.
1975: Rain until afternoon, starting again in the evening.
1976: Heavy rain until evening, starting again overnight.
1977: Mostly cloudy throughout the day.
1978: Mostly cloudy throughout the day.
1979: Breezy in the morning and mostly cloudy throughout the
day.
1980: Mostly cloudy starting in the afternoon.
1981: Rain overnight.
1982: Mostly cloudy throughout the day.
1983: Mostly cloudy in the morning.
1984: Mostly cloudy throughout the day.
1985: Rain until afternoon.
1986: Mostly cloudy throughout the day.
1987: Light rain starting in the afternoon, continuing until
evening.
1988: Partly cloudy throughout the day.
1989: Partly cloudy until evening.
1990: Heavy rain until afternoon.
1991: Mostly cloudy throughout the day.
1992: Mostly cloudy throughout the day.
1993: Partly cloudy throughout the day.
1994: Partly cloudy until evening.
1995: Partly cloudy overnight.
```

```
1996: Foggy in the morning.
1997: Rain overnight.
1998: Partly cloudy until evening.
1999: Mostly cloudy throughout the day.
2000: Mostly cloudy starting in the afternoon, continuing until
evening.
2001: Partly cloudy until afternoon.
2002: Partly cloudy starting in the afternoon.
2003: Rain in the morning and afternoon.
2004: Mostly cloudy starting in the afternoon.
2005: Partly cloudy in the morning.
2006: Partly cloudy throughout the day.
2007: Rain in the morning.
2008: Heavy rain in the evening.
2009: Partly cloudy starting in the afternoon, continuing until
evening.
2010: Partly cloudy throughout the day.
2011: Rain starting in the afternoon, continuing until evening.
2012: Partly cloudy throughout the day.
2013: Light rain in the morning.
2014: Partly cloudy starting in the afternoon, continuing until
evening.
2015: Partly cloudy in the afternoon.
2016: Partly cloudy until evening.

fog                :
partly-cloudy-day  :******************************
partly-cloudy-night :**
rain               :***********
wind               :
```

I threw in the last part for fun. It is a graph of the icons a weather program might show (another kind of summary, really) for the day. Looks like I've lived a pretty "partly cloudy" life. I'm curious to see what sort of fun things you can do with this API, so have at it.

Take care, and I'll see you next time.