In this assignment you will write a Ruby program that uses HTTParty to call a remote service.

**JSON**

You need to understand the basics of JSON (JavaScript Object Notation). Although the J in JSON stands for JavaScript, JSON is a lightweight and easy way to represent structured data (compared to alternatives such as XML) using any programming language.

If you are not familiar with JSON data format read

<https://www.tutorialspoint.com/json/index.htm> and the topics on JSON Overview, Syntax and Datatypes.

Read the page on JSON with Ruby <https://www.tutorialspoint.com/json/json_ruby_example.htm> and understand the example of how JSON data is represented in Ruby using hashes and arrays.

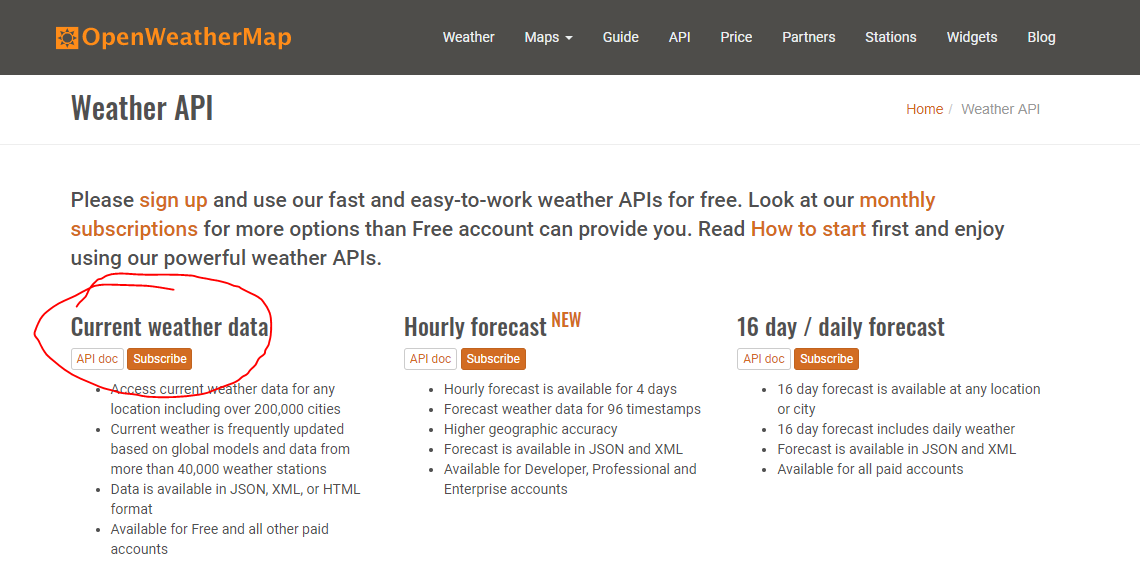
**Weather api**

We will use the weather service available at

<http://api.openweathermap.org>

To use the service, you will need an api key. The purpose of the api key is to control access to the site and to limit the number of requests you can do per day for a free subscription.

1. Go to the openweathermap site



1. click on Subscribe for Current weather data and obtain a free api key. It will look something like

' b6907d289e10d714a6e88b30761fae22'

1. Install the ruby gem ‘httparty’ using the command

$ gem install httparty

1. Create a directory for this assignment

$ mkdir assignment5

$ cd assignment5

Download the zip file this assignment and unzip into this directory.

You will have files

* city\_weather.rb
* README.docs
* README.pdf
* solution.rb

1. Edit the file city\_weather.rb and add statements for base\_uri, default\_params, format and define the CityWeather.for class method.
2. require 'httparty'
3. class CityWeather
4. include HTTParty
5. # default\_options.update(verify: false) # Turn off SSL
6. base\_uri "http://api.openweathermap.org/data/2.5/weather"
7. default\_params appid: 'b6907d289e10d714a6e8b308761fae22'
8. format :json
9. def CityWeather.for(city\_name)
10. get('', query: { q: city\_name })
11. end
12. end

In line 5 we are using HTTParty as a ***mixin*** to the class CityWeather. This makes all the methods of HTTParty module available as if they were methods in HTTParty. Similar to inheritance in Java, but without making CityWeather a subclass of HTTParty. Read more about ***mixin*** in chapter 3.7 in the textbook.

Update line 9 with the api key that you obtained from step 2. HTTParty will construct a URL using base\_uri, default\_params and query parameters from the GET statement (line 13). HTTPary then makes a HTTP GET request on line 134.

Line 10 specifies that the HTTP request header will specify that data be returned in JSON format

Content-Type: application/json

Accept: application/json

Line 12 defines a class method (a static method in Java).

Line 13 will build and execute an HTTP GET request. The first parameter is empty string because there is nothing that needs to be appended to the base\_uri. The query: parameter indicates that parameters will be passed in the URL The parameter in this case is q=<city\_name>.

The complete URL that HTTParty will build will look something like

<https://api.openweathermap.org/data/2.5/weather?q=London&appid=b6907d289e10d714a6e88b30761fae22>

1. HTTParty will take the JSON data returned by openweathermap and convert to a Ruby data structure of arrays and hashes.

The JSON data returned from the above URL in the body of the HTTP response will look like

{

"coord":{

"lon":-0.13,

"lat":51.51

},

"weather":[

{

"id":300,

"main":"Drizzle",

"description":"light intensity drizzle",

"icon":"09d"

}

],

"base":"stations",

"main":{

"temp":280.32,

"pressure":1012,

"humidity":81,

"temp\_min":279.15,

"temp\_max":281.15

},

"visibility":10000,

"wind":{

"speed":4.1,

"deg":80},

"clouds":{

"all":90

},

"dt":1485789600,

"sys":{

"type":1,

"id":5091,

"message":0.0103,

"country":"GB",

"sunrise":1485762037,

"sunset":1485794875

},

"id":2643743,

"name":"London",

"cod":200

}

HTTParty will convert the JSON string data into a Ruby hash of this data with keys for

* “coord”,
* “weather”,
* “base”,
* “main”,
* “visibility” and so on.

The value of “coord” will be another hash with keys “lon” and “lat”

{ “coord” => { “lon” => -0.13, “lat” => 51.51} , “weather” => { . . . } , “base” => “stations”, . . . }

The “weather” key has a value that is an array of length 1 containing a hash. Data within braces { } are turned into a ruby hash. Data within brackets [ ] are converted into a ruby array.

To retrieve the current temp for London we would do the following

data = CityWeather.for(‘London’)

data[‘main’][‘temp’]

data[‘main’] retrieves the hash consisting of

{

"temp" => 280.32,

"pressure" => 1012,

"humidity" => 81,

"temp\_min" => 279.15,

"temp\_max" => 281.15

}

[‘temp’] then retrieves the temp value of 280.32 from this nested hash.

1. To test out CityWeather, run the solution.rb file with the command

$ ruby solution.rb

1. What does openweathermap.org return if the city name is misspelled? Run solution.rb and enter a city name of bad\_city and see what is returned.
2. Temperature values are in degrees Kelvin (not Fahrenheit or Celsius). Wind speed is in meters/seconds. Timestamp values are in UTC UNIX time units.

Study the code in solution.rb that converts temp value to Fahrenheit and converts time values to displayable date time strings.

**Submit your city\_weather.rb file to iLearn to get credit for this assignment.**

**Next up Assignment 6**

* We will create our first Rails application that retrieves information about cities. Some of the data will come from a local database, but the current weather for the city will be retrieved using code from the CityWeather class.