

Lab 4 Part 1

Calling APIs

Overview

In this lab, we will look at:

- Using a weather API
- Consuming an API (make a call to an API)

This lab will help you to gain an understanding of using an application API. You will use the OpenWeatherMap API.

You will also learn how to access an API from your Node.js application program.

Tasks

Task 1: Using the OpenWeatherMap API

1. To use the open weather map, sign up at <http://openweathermap.org/appid>

2. Then you can get an API key from https://home.openweathermap.org/api_keys

The data returned by the 'current weather API' is described at <https://openweathermap.org/current>

Note that according to the docs "Activation of an API key for Free and Startup accounts **takes 10 minutes**"

3. In your browser, visit the link below:

`api.openweathermap.org/data/2.5/weather?q=London,uk&APPID=YourAPIKey`

Replace **YourAPIKey** with your API key.

You should see some data come back, in JSON format:

```
{
  - coord: {
    lon: -0.1257,
    lat: 51.5085
  },
  - weather: [
    - {
      id: 801,
      main: "Clouds",
      description: "few clouds",
      icon: "02d"
    }
  ],
  base: "stations",
  - main: {
    temp: 286.27,
    feels_like: 285.86,
    temp_min: 285.07,
    temp_max: 287.24,
    pressure: 996,
    humidity: 85
  },
  visibility: 10000,
  - wind: {
    speed: 5.66,
    deg: 210,
    gust: 10.8
  },
  - clouds: {
    all: 20
  },
  dt: 1667898931,
  - sys: {
    type: 2,
    id: 2075535,
    country: "GB",
    sunrise: 1667891164,
    sunset: 1667924547
  },
  timezone: 0,
  id: 2643743,
  name: "London",
  cod: 200
}
```

4. Now change the above URL to see weather information related to another city, say Manchester.

Task 2: Add a weather forecast to Berties Books

Berties books is a bit of an odd website. As well as selling books, they will provide weather forecasts! Let's add a weather forecast page to our site, using the OpenWeatherData API.

You can continue working on your previous application code, i.e. from the last lab. If you have set up a local development environment you can complete this lab entirely in this local environment. There is no need to work on the server.

5. Test your Berties Books application to make sure it still works
6. Create a new route in your web application called '/weather' using a 'get' route handler.
7. Install the request module. This will allow us to make HTTP requests to an API and get the result back.

Note that the request module uses an old usage pattern and is being replaced by more modern approaches. Read [this article](#) for more information. But we will continue to use request for simplicity.

```
npm install request
```

8. then add this piece of code to your new route (/weather) to make a call to the weather api and get today's weather in London:

```
const request = require('request');

let apiKey = '*****';
let city = 'london';
let url =
`http://api.openweathermap.org/data/2.5/weather?q=${city}&units=metric&appid=${
apiKey}`

request(url, function (err, response, body) {
  if(err){
    console.log('error:', error);
  } else {
    res.send(body);
  }
});
```

Replace '*****' in the above code with your API Key.

9. Run the application and navigate to the /weather route. You should see the weather in JSON format (you can look out the window to see the weather in “real life” format).

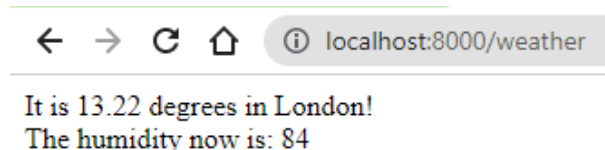
Task 3: Make the weather forecast more human-friendly

Json format is great, but not so easy on the eye. Let's make it more readable:

10. Comment out `res.send(body)` and add the following lines:

```
var weather = JSON.parse(body)
var wmsg = 'It is ' + weather.main.temp +
  ' degrees in ' + weather.name +
  '!' <br> The humidity now is: ' +
  weather.main.humidity;
res.send (wmsg);
```

11. Now run your web application and access weather route. You should see something like this:



The screenshot shows a web browser window with the address bar displaying 'localhost:8000/weather'. The page content shows the weather in London: 'It is 13.22 degrees in London!' and 'The humidity now is: 84'.

- 12.** Add a link to your weather route from your home page.

Task 4: Add some interactivity

- 13.** Add some interactivity to your 'weather' route by adding a form with an edit box for user to enter the city name and then display weather information related to that city.

Task 5: Additional weather info

- 14.** Can you display other information related to weather such as wind and etc?
- 15.** Can you organise the page in a nicer format, using better font size and colours?

Task 6: Add some error handling

- 16.** Run your application and browse to the API page. Enter a non-existent place name. Your program will crash!

Add some error handling to deal with this. Here is an example of how you might do this:

```
var weather = JSON.parse(body)
if (weather!==undefined && weather.main!==undefined) {
  var wmsg = 'It is ' + weather.main.temp +
    ' degrees in ' + weather.name +
    '! <br> The humidity now is: ' +
    weather.main.humidity;
  res.send (wmsg);
}
else {
  res.send ("No data found");
}
```

This error handling is **really important!** You can't guarantee that APIs will return results in the exact format you expect or indeed return results at all. If you don't do this correctly your app may crash and we won't be able to complete the marking!

END