Call an upstream API to get data using RestTemplate

Get Real

In this chapter, we will get a real world example.

we will show the following things:

- How to set a path parameter, as opposed to a query string.
- How to get weather data from a third party system using APIs.
- How to use Spring RestTemplate.
- How to get configuration value from application.properties.
- How to use Java Jackson to deserialize Json data into Java Objects.

Controller

We add a new method in our HelloController

```
@GetMapping("/hello/area/{city}")
public WeatherModel GetAreaDetails(@PathVariable String city)  {

    RestTemplate restTemplate = new RestTemplate();

    WeatherModel weather = restTemplate.getForObject(weatherPath+ city, WeatherModel.class);
    return weather;
}
```

Notice the GetMapping path. It's now 'hello/area/{**city**} which is how we declared a path variable, as you can see it defined in the line below as a string.

Also, see the use of the RestTemplate to populate the response object WeatheModel.

In Spring Boot, you use RestTemplate to consume web services over HTTP methods.

Finally as you can see, we are capturing our upstream API response in WeatherModel and returning it as response.

At this point, there are two things you should wonder.

One, the variable weatherPath in,line 4, where is it defined, how does it get it's value

Two, where and how we defined WeatherModel.

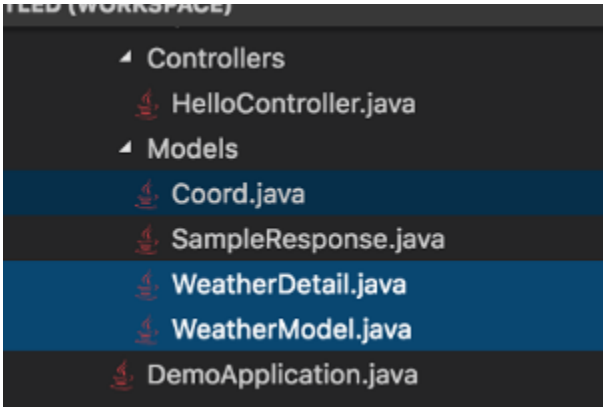
WeatherPath is where we have configured the Url for the Upstream Weather API in application.properties and on line 4, we are appending the city that has been passed from the path.

```
weather.path = https://samples.openweathermap.org/data/2.5/weather?
appid=b6907d289e10d714a6e88b30761fae22&q=
```

And then we capture it's value in weatherPath as below

```
@Value("${weather.path}")
private String weatherPath;
```

Finally, WeatherModel is the representation of a nested Json returning the weather and coordinates for a given city.



In this example, weatherModel is the parent class which has one instance of Coord and an array of WeatherDetail.

```
package com.rbs.sample.Models;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@JsonIgnoreProperties(ignoreUnknown = true)
public class WeatherModel
{
    private String id;

    private String dt;
```

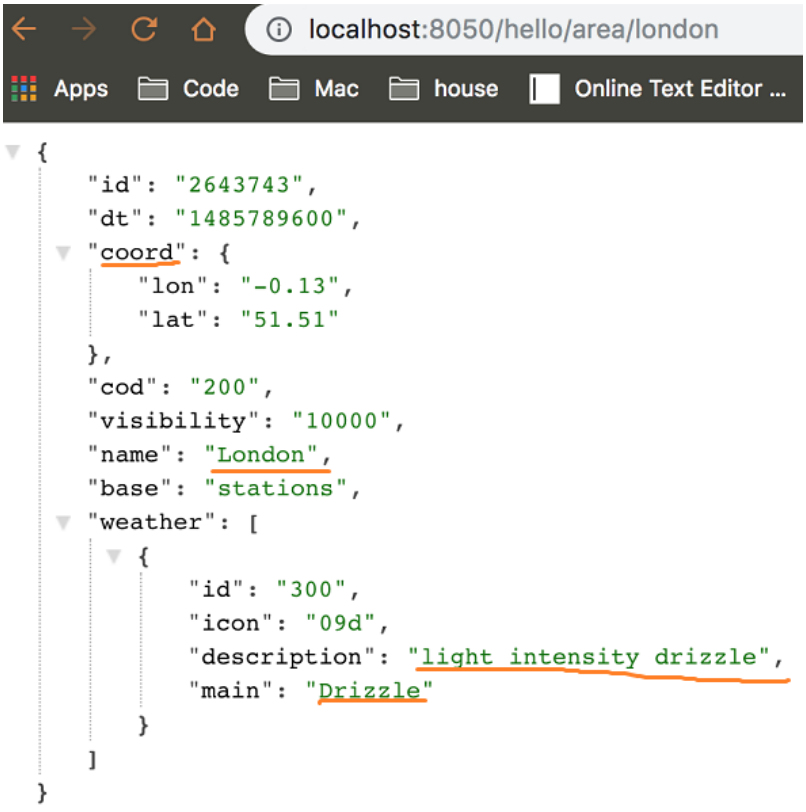
Note, the yellow underlined areas.

The first one is we are importing Jackson libraries for serialize and de-serialize Json to Java objects and vice-versa.

The second one is telling our compiler to ignore the properties that are part of the json response, but not defined in the class. This is particularly handy when you only need to pick a smaller subsection of a large response.

> All code for this sample will be available in gitlab. So don't worry that you don't have the full model here.

Finally, lets try running our api and call the new url we just declared to get the weather and cordinates for London, GB.



As you can see, London is rainy today. (nothing new there)