# 4CS017 – Internet Software Architecture tutorial

## Further PHP and MySQL - “Chain of Responsibility” pattern

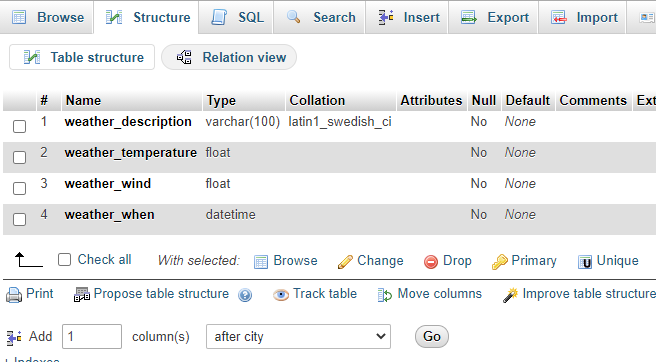
*What will you learn today?*

You will produce a simple implementation of the “Chain of Responsibility” pattern (see lecture slides), in order to retrieve data from the OpenWeather API **when required**.

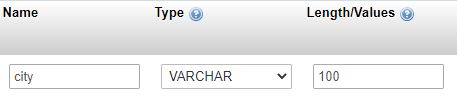
## Part 1 – Adding a “city” parameter to your API

Your API is a bit boring at the moment, in that it only returns data for a given city. Let’s add a “city” parameter to it, just like the OpenWeather API!

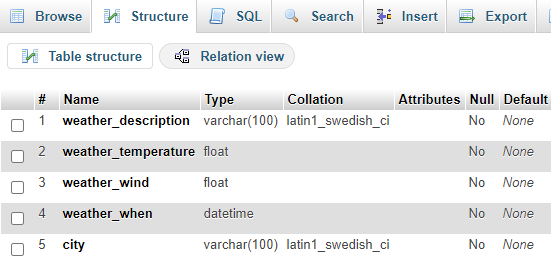
1. First, add a “city” field to your existing “weather” table:



Specify the following field name and data type:



If all goes well, you should see something like this under the “Structure” tab:



1. Next, amend the SQL query in your “my-api.php” file from last week, so that you only select weather data for a given city. While we are here, let’s only select “fresh” data, i.e. with a date no older than x units of time:

$sql = "SELECT \*

FROM weather

WHERE city = '{$\_GET['city']}'

AND weather\_when >= DATE\_SUB(NOW(), INTERVAL 10 SECOND)

ORDER BY weather\_when DESC limit 1";

$result = $mysqli -> query($sql);

A couple of comments on the code:

* + $\_GET['city'] looks for a parameter called “city” in your URL. You can now pass a value to your API like this:
    1. https://mi-linux.wlv.ac.uk/~in9352/weather/my-api.php?city=london
  + Don’t worry too much about the SQL syntax for checking the date... but you might want to amend the actual interval. I’ve set it to 10 seconds here so that it’s easy to demonstrate. More info [here](https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html#function_date-add).

1. Finally, you can now pass the city from your JavaScript to your API, just like we did with the OpenWeatherMap API. Update the “fetch” statement in your JavaScript, like this:

fetch('https://mi-linux.wlv.ac.uk/~in9352/weather/my-api.php?city=london')

*Note: this is my URL!*

1. Checkpoint: Your app should be functional, **but only** if you’ve inserted “fresh” (less than 10 seconds old) data for **your city** in your MySQL table.

## Part 2 – Importing fresh data from the OpenWeather API

Okay, on to the interesting bit now. Let’s implement the “Chain of Responsibility” pattern, i.e. your API will return fresh data if available, or get it from the OpenWeather API if not!

1. Using Notepad++ or any text editor of your choice, create a **new** file called “data-import.php” or anything similar.
   * **Important**: you can call it anything you like, but it **needs to have the .php extension**, or else it won’t run. Also please get into a habit of using **lowercase** file names with **no spaces**. Once finished, upload the file to mi-linux, as per previous instructions.
2. Type in (or copy / paste / review) the following code:

<?php

// Select weather data for given parameters

$sql = "SELECT \*

FROM weather

WHERE city = '{$\_GET['city']}'

AND weather\_when >= DATE\_SUB(NOW(), INTERVAL 10 SECOND)

ORDER BY weather\_when DESC limit 1";

$result = $mysqli -> query($sql);

// If 0 record found

if ($result->num\_rows == 0) {

$url = 'https://api.openweathermap.org/data/2.5/weather?q=' . $\_GET['city'] . '&appid=xxx&units=metric';

// Get data from openweathermap and store in JSON object

$data = file\_get\_contents($url);

$json = json\_decode($data, true);

// Fetch required fields

$weather\_description = $json['weather'][0]['description'];

$weather\_temperature = $json['main']['temp'];

$weather\_wind = $json['wind']['speed'];

$weather\_when = date("Y-m-d H:i:s"); // now

$city = $json['name'];

// Build INSERT SQL statement

$sql = "INSERT INTO weather (weather\_description, weather\_temperature, weather\_wind, weather\_when, city)

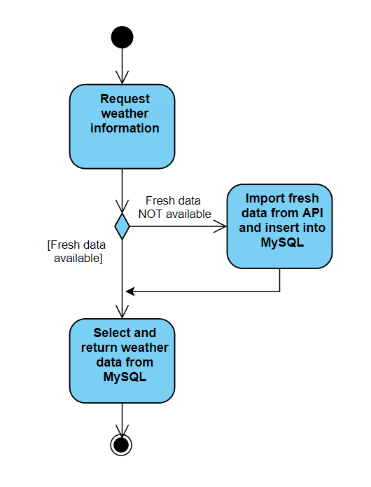
VALUES('{$weather\_description}', {$weather\_temperature}, {$weather\_wind}, '{$weather\_when}', '{$city}')";

// Run SQL statement

$mysqli -> query($sql);

}

?>



Comments on the code:

* The first block runs the same SQL query as your API, in order to **count** how many records are returned. If 0, it makes the decision to **import fresh data from the OpenWeather API**. In order words, this IF statement represent the decision node on your **Activity Diagram**!
* The next block of code **retrieves data from the OpenWeather API**, and as such looks quite similar to our JS from Prototype 1.
* Finally, once we’ve got the fresh data, instead of displaying it on the screen, **we INSERT it into our database table** using SQL. Obviously, you’ll have to **tailor the query according to your table name and field names**. Please note that text and date values have ' ' around them.

1. **Finally,** you need to call the code above in your main “my-api.php”file. You can do this by simply “including” the import script, here:

// Connect to database

…

// First, check requested data is present and fresh

include('data-import.php');

// Select weather data for given parameters

…

**All done**. Here is mine:

<https://mi-linux.wlv.ac.uk/~in9352/weather/task2-client.html>

If you refresh the page **every second**, you will notice that every 10 seconds (ish), the refresh will take a bit longer (we’ll talk about performance soon), and the “date updated” will change… my API has **just fetched fresher data from the OpenWeather API**! Obviously in real life you would set this to a longer interval (maybe 1 hour?)

## Part 4 – Going further (important: for fun - **not** required for the assessment)

*“I have finished all the work above, what shall I do next?”*

1. Try and implement a “search city” button (I demonstrated this during the surgery in week 4 – about 33:30 in the [video](https://wlv.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=3a972a2d-eff8-4e76-9ed4-acd301219ea2)). Then try to ask for weather information for various cities around the globe and see how your database “cache” is slowly filling up with data:

