# 4CS017 – Internet Software Architecture tutorial

## PHP and MySQL

*What will you learn today?*

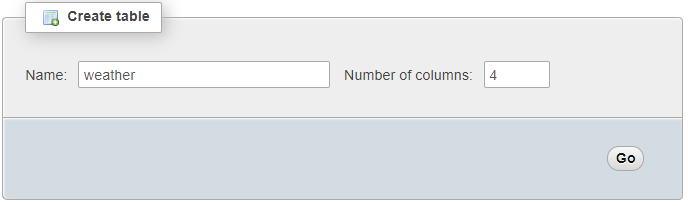
You will learn to connect to a MySQL database from PHP, retrieve some data, and display the data as JSON (aka, you will write a simple REST API).

Important: there is a checkpoint at the end of each part, to ensure that you’ve done the work correctly so far before moving on.

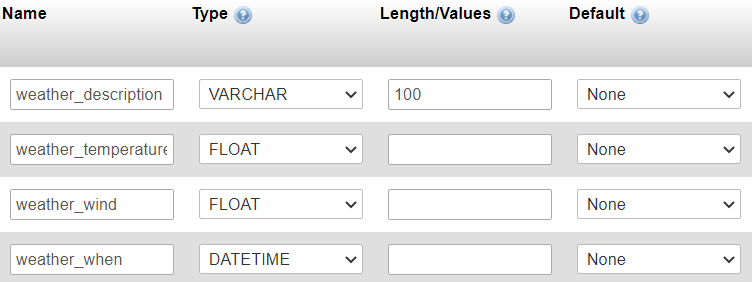
## Part 1 – Creating your MySQL database table

First, let’s create a simple database **table** in MySQL, with some dummy weather data.

1. Register a MySQL account on mi-linux - **Please ignore this step if you already have a MySQL account on mi-linux**.
   * Connect to our facilities website: <https://mi-linux.wlv.ac.uk/facilities/>
   * Choose to “register” for MySQL, read the terms and confirm.
   * **Important**: Write down/copy and paste/screenshot your credentials.
   * Wait for a few minutes for your account to be validated.
2. Connect to your MySQL database via a web frontend called phpMyAdmin, here:
   * <https://mi-linux.wlv.ac.uk/phpmyadmin/>
   * Use your credentials from step 1 above.
   * If you have lost or forgotten your MySQL password, you can reset it on the facilities website (see URL in step 1)
3. Once connected, **select** your database on the left (you only get the one, called “db” + your student number), then **create a table** called “weather” with 4 fields, as follow:

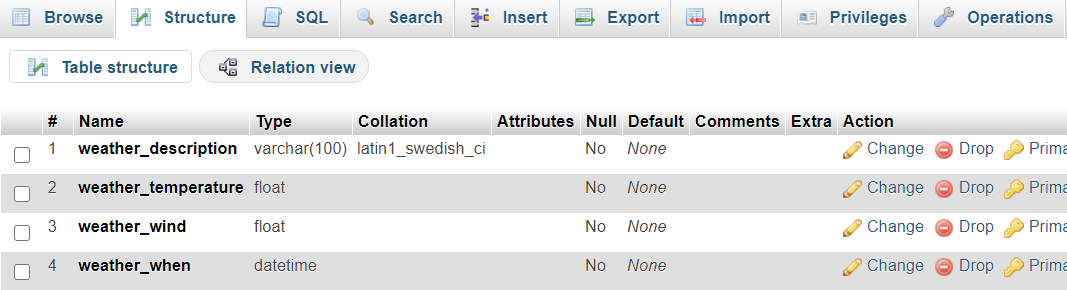


Specify the following field names and data types:

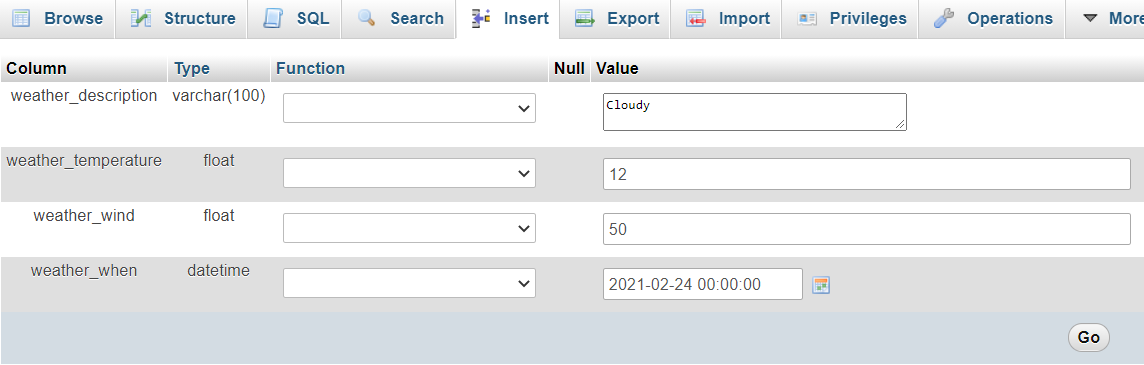


**Important**: always choose a suitable [data type](https://www.w3schools.com/sql/sql_datatypes.asp) (varchar for text, int for whole numbers, float for numbers with decimals, datetime for dates with a time etc.)

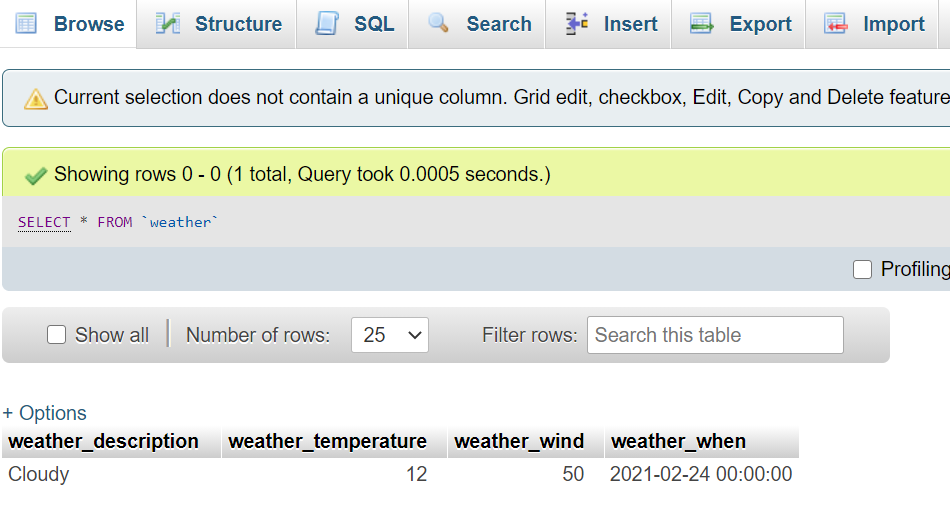
If all goes well, you should see something like this under the “Structure” tab:



1. Next week we will learn to import real data into our table from the OpenWeatherMap API, but for now **let’s just insert some dummy data**. Click on the “Insert” tab, then type in some weather data as follow and press “Go”:



1. Checkpoint: You should be able to see your data in the “Browse” tab.



## Part 2 – Accessing your database table from PHP

Now let’s use PHP to connect to the database table above, retrieve the data and display it as JSON.

1. Using Notepad++ or any text editor of your choice, create a new file called “my-api.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**.
2. Type in (or copy / paste / review) the following code:

<?php

// Connect to database

$mysqli = new mysqli("localhost","your\_mysql\_user","your\_mysql\_pasword","your\_database\_name");

if ($mysqli -> connect\_errno) {

echo "Failed to connect to MySQL: " . $mysqli -> connect\_error;

exit();

}

// Execute SQL query

$sql = "SELECT \*

FROM weather

ORDER BY weather\_when DESC limit 1";

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

// Get data, convert to JSON and print

$row = $result -> fetch\_assoc();

print json\_encode($row);

// Free result set and close connection

$result -> free\_result();

$mysqli -> close();

?>

Comments on the code:

* The first block connects to the database server and selects your database. Make sure you use your MySQL logon and passwords from part 1. The database name is “db” + your student number, NOT your table name.
* The second block runs an SQL statement that selects the **last** record in the “weather” table (I know we only have one record in there for now, but eventually we will have more, and we will always want to select the latest weather data)
* The third block converts the data to JSON and prints it (i.e. sends it back to the browser to be displayed)
* The last block does some tidying up.

1. **Publish** your script on mi-linux as per last week’s instructions (via SFTP) – don’t forget to set the permissions to public.
2. Checkpoint: **Browse** to your file in Chrome, to make sure it works (make sure you use your own student number and file name):

'https://mi-linux.wlv.ac.uk/~0123456/my-api.php'

It should display the data from your database table in JSON format, a bit like this:

{"weather\_description":"Cloudy","weather\_temperature":"12","weather\_wind":"50","weather\_when":"2021-02-24 00:00:00"}

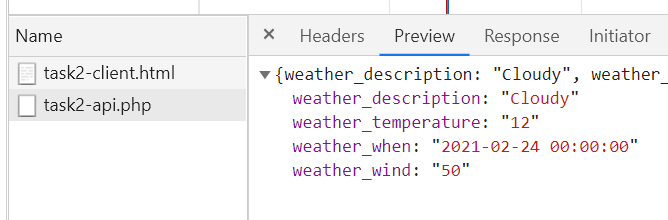
## Part 3 – Modifying your client-side app

Now all we need to do is **point** your existing client-side JavaScript app to your newly created PHP API, and update some of the fields in the JSON. Make a copy of your prototype1.html file called **prototype2.html**, and make the following changes:

1. Update the **fetch** statement to point to your URL from Part 2 (make sure you use your own student number and file name) rather than the OpenWeatherMap API:

fetch('https://mi-linux.wlv.ac.uk/~0123456/my-api.php')

1. Our JSON message is now structured differently, as per our database fields in part 2… look in the Chrome Developer Tools:



So, update the field names in the part of the script that reads the JSON response, like this:

document.getElementById("myWeather").innerHTML = response.weather\_description;

document.getElementById("myTemperature").innerHTML = response.weather\_temperature;

(Note: the IDs of your HTML elements might be different)

1. **Upload** your client app to mi-linux, as per last week’s instructions.

Checkpoint: All done! **Browse** to your client app, and it should function just like before… no change in appearance, but it’s a different story “under the hood”, as it is now fetching the data from YOUR server-side PHP API:

* It will still work if the OpenWeather API goes down (the data just won’t be as fresh).
* It won’t hit the API as often, and as such won’t incur a cost.
* Here is [mine](https://mi-linux.wlv.ac.uk/~in9352/weather/task2-client.html), as a working example (it looks very simple looks-wise!)

## 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. Create more weather fields (e.g. humidity etc.) in your database table and amend PHP + JavaScript accordingly.
2. **More challenging**: Currently our API always returns the same data. You could improve it by allowing it to accept parameters, and return different data depending on the value of the parameters provided (just like the OpenWeather API).

You will need to use the $\_GET array to retrieve values from the URL, and a different SQL statement depending in the value provided.