

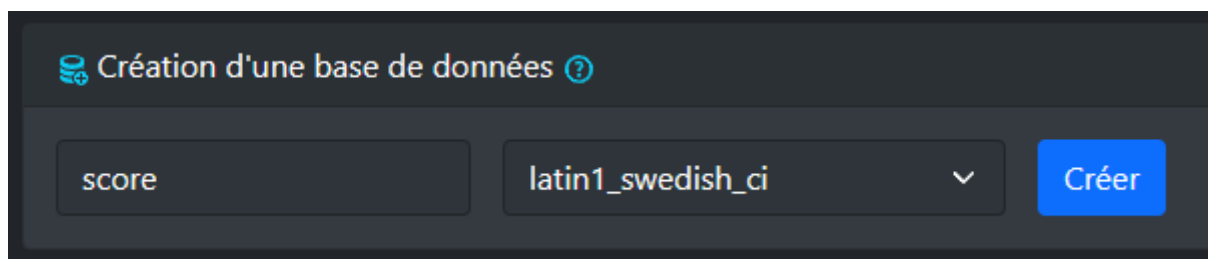
How to connect your Clickteam application to an online database for a scoreboard for example

Things we will need:

- A PHP web server with mysql (I'm using 000webhost)
- The Clickteam MFA file from the zip file as example
- The PHP script from the zip file
- A text editing software (I'm using Visual Studio Code)

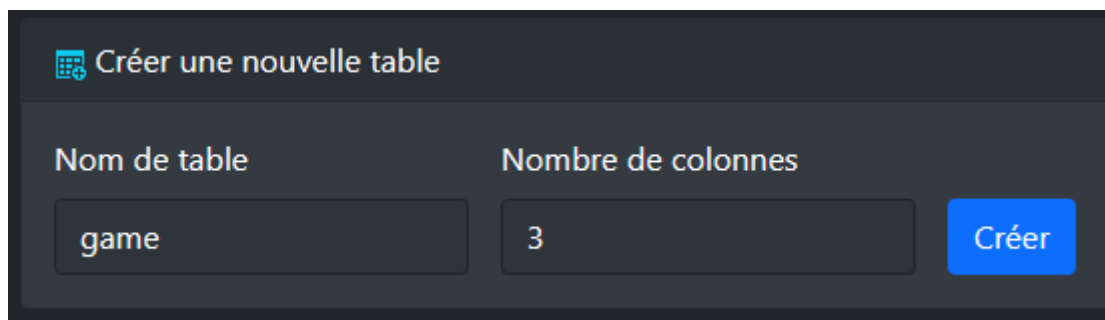
Creating the database:

Open your database managing tool (in my case it's phpMyAdmin) and create a data base with the name of your choice.



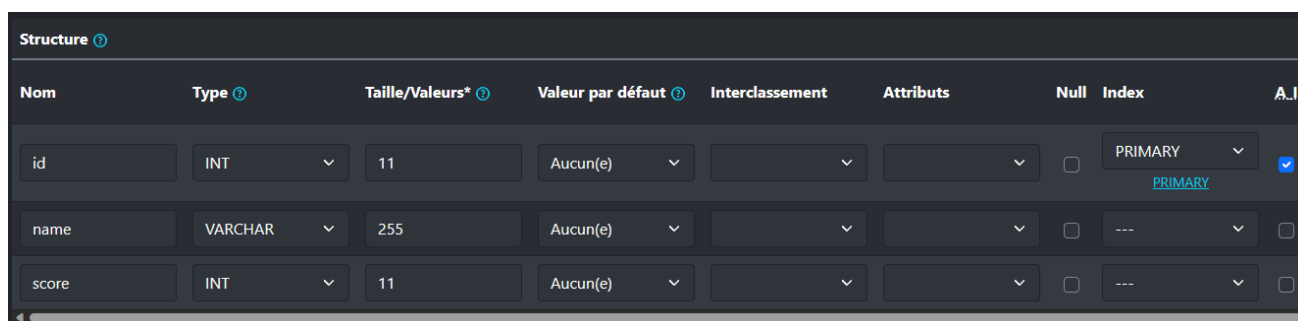
The screenshot shows the 'Création d'une base de données' (Create a new database) interface in phpMyAdmin. It features a text input field containing 'score', a dropdown menu set to 'latin1_swedish_ci', and a blue 'Créer' (Create) button.

Then create a table, I will create one for a game with an online scoreboard, so I'll name it to the name of my game.



The screenshot shows the 'Créer une nouvelle table' (Create a new table) interface in phpMyAdmin. It has two input fields: 'Nom de table' (Table name) with 'game' and 'Nombre de colonnes' (Number of columns) with '3'. A blue 'Créer' (Create) button is on the right.

Then set 3 elements, an ID, a name and a score. The only 3 things I need for my game. Put as many elements as you and your application needs.



The screenshot shows the 'Structure' (Structure) tab in phpMyAdmin, displaying the table structure for the 'game' table. The table has three columns: 'id', 'name', and 'score'.

Nom	Type	Taille/Valeurs*	Valeur par défaut	Interclassement	Attributs	Null	Index	A.I
id	INT	11	Aucun(e)			☐	PRIMARY	☒
name	VARCHAR	255	Aucun(e)			☐	---	☐
score	INT	11	Aucun(e)			☐	---	☐

Setting up the PHP script:

Open the script file with your text editing software of choice and first, we're gonna look at this block of code, it for the logins for our database.

```
$host = "";  
$database = "";  
$login = "";  
$password = "";
```

You'll have to put between the quotes, the url of where your database is hosted, by default it's localhost (in most cases), then the name of the database, then the login of your database managing tool (in my case it's phpMyAdmin) and it's password, if there's none, let the variable empty.

```
$host = "localhost";  
$database = "score";  
$login = "root";  
$password = "";
```

Now we're gonna look at this giant block of code.

```
if(isset($_POST['player']) && isset($_POST['score'])) {  
    $requete = "SELECT COUNT(*) FROM game WHERE player = '{$_POST['player']}'";  
    $resultat = $cnn->query($requete) or die(print_r($database->errorInfo()));  
    while($row = $resultat->fetch()){  
        if($row['COUNT(*)'] == 1) {  
            $requete1 = "UPDATE game SET score = '{$_POST['score']}' WHERE player = '{$_POST['player']}'";  
            $cnn->exec($requete1) or die(print_r($database->errorInfo()));  
            $cnn=null;  
        }  
        else {  
            $requete1 = "INSERT INTO game (player, score)  
VALUES ('{$_POST['player']}','{$_POST['score']}')";  
            $cnn->exec($requete1) or die(print_r($database->errorInfo()));  
            $cnn=null;  
        }  
    }  
}
```

First, we check if two POST variable exists, one for the player and another one for the score, it's for security purposes. The name of those variables will be set in the Clickteam application so you can already name them. The rest of the code is here to put or edit things on/from the database. If your table has more element (other than the ID) put those in the code. For example first in the if statement to verify that the thing we will add are all here when doing the web request.

```
if(isset($_POST['first_name']) && isset($_POST['last_name']) && isset($_POST['city']) && isset($_POST['phone'])) {
```

Then this line is for grabbing the number of rows the table have with the name of our player that we put in the Clickteam application and the first if statement.

```
$requete = "SELECT COUNT(*) FROM game WHERE player = '{$_POST['player']}'";
```

It's to check if for example we already publish an online score with the same name and want to update the score instead of re-adding one with the same name. If you don't need this part I write the exact same thing but without this verification and editing thing as comment, select the one you need.

```
/* The same code as above but without the checking and editing section, if there's a new entry with the same name it will still add it without overwriting the previous one */  
  
if(isset($_POST['player']) && isset($_POST['score'])) {  
    $requete = "INSERT INTO game (player, score)  
    VALUES ('".$_POST['player']."', '".$_POST['score']."')";  
    $cnn->exec($requete) or die(print_r($database->errorInfo()));  
    $cnn=null;  
}  
*/
```

The rest of the code is to check with our name is already in the database and overwriting the row containing our name with the new score.

```
if($row['COUNT(*)'] == 1) {  
    $requete1 = "UPDATE game SET score = '".$_POST['score']."' WHERE player = '".$_POST['player']."'";  
    $cnn->exec($requete1) or die(print_r($database->errorInfo()));  
    $cnn=null;  
}
```

Or to let it create an entry if the name isn't already in the database.

```
else {  
    $requete1 = "INSERT INTO game (player, score)  
    VALUES ('".$_POST['player']."', '".$_POST['score']."')";  
    $cnn->exec($requete1) or die(print_r($database->errorInfo()));  
    $cnn=null;  
}
```

Of course, change every names and variables as you need. In our example with the first if statement, the rest of the block of code will look like this.

```
$requete = "SELECT COUNT(*) FROM game WHERE first_name = '".$_POST['name']."'";  
$resultat = $cnn->query($requete) or die(print_r($database->errorInfo()));  
while($row = $resultat->fetch()){  
    if($row['COUNT(*)'] == 1) {  
        $requete1 = "UPDATE game SET first_name = '".$_POST['first_name']."', last_name = '".$_POST['last_name']."', city = '".$_POST['city']."', phone = '".$_POST['phone']."' WHERE first_name = '".$_POST['first_name']."'";  
        $cnn->exec($requete1) or die(print_r($database->errorInfo()));  
        $cnn=null;  
    }  
    else {  
        $requete1 = "INSERT INTO game (first_name, last_name, city, phone)  
        VALUES ('".$_POST['first_name']."', '".$_POST['last_name']."', '".$_POST['city']."', '".$_POST['phone']."')";  
        $cnn->exec($requete1) or die(print_r($database->errorInfo()));  
        $cnn=null;  
    }  
}
```

And finally we're gonna look at this block of code.

```
if(isset($_GET['leaderboard'])) {  
    $scores = "";  
  
    $requete = "SELECT * FROM game ORDER BY score DESC";  
    $resultat = $cnn->query($requete) or die(print_r($database->errorInfo()));  
    while($row = $resultat->fetch()){  
        $scores .= "{$row['name']} - {$row['score']}|";  
    }  
  
    echo substr($scores, 0, -1);  
}
```

The first if statement is to check if a GET variable exists, to show the score in our application or a web page, then we initialize a scores variable.

```
$scores = "";
```

Then select everything from the table of our game in a descending order to show the highest score above all the others and set all of them in a variable.

```
$requete = "SELECT * FROM game ORDER BY score DESC";  
$resultat = $cnn->query($requete) or die(print_r($database->errorInfo()));  
while($row = $resultat->fetch()){  
    $scores .= "{$row['name']} - {$row['score']}|";  
}
```

At the end we put the character "|" to tell to our application when an entry ends, to put another in a new line. I call this a delimiter. But at the end, there will be a last delimiter and the application will understand this as a last entry even if there's nothing after the "|". So to prevent this in the same function to show all of the scores, we just remove the last character with the substr PHP function.

```
echo substr($scores, 0, -1);
```

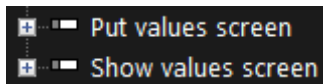
The -1 at the end is to specify the number of character we will remove at the end. Still, change the values and variables as you need, in our example the block of code will look like this.

```
if(isset($_GET['leaderboard'])) {  
    $entries = "";  
  
    $requete = "SELECT * FROM game ORDER BY first_name DESC";  
    $resultat = $cnn->query($requete) or die(print_r($database->errorInfo()));  
    while($row = $resultat->fetch()){  
        $entries .= "{$row['first_name']} - {$row['last_name']} - {$row['city']} - {$row['phone']}|";  
    }  
  
    echo substr($entries, 0, -1);  
}
```

Setting up you application on Clickteam:

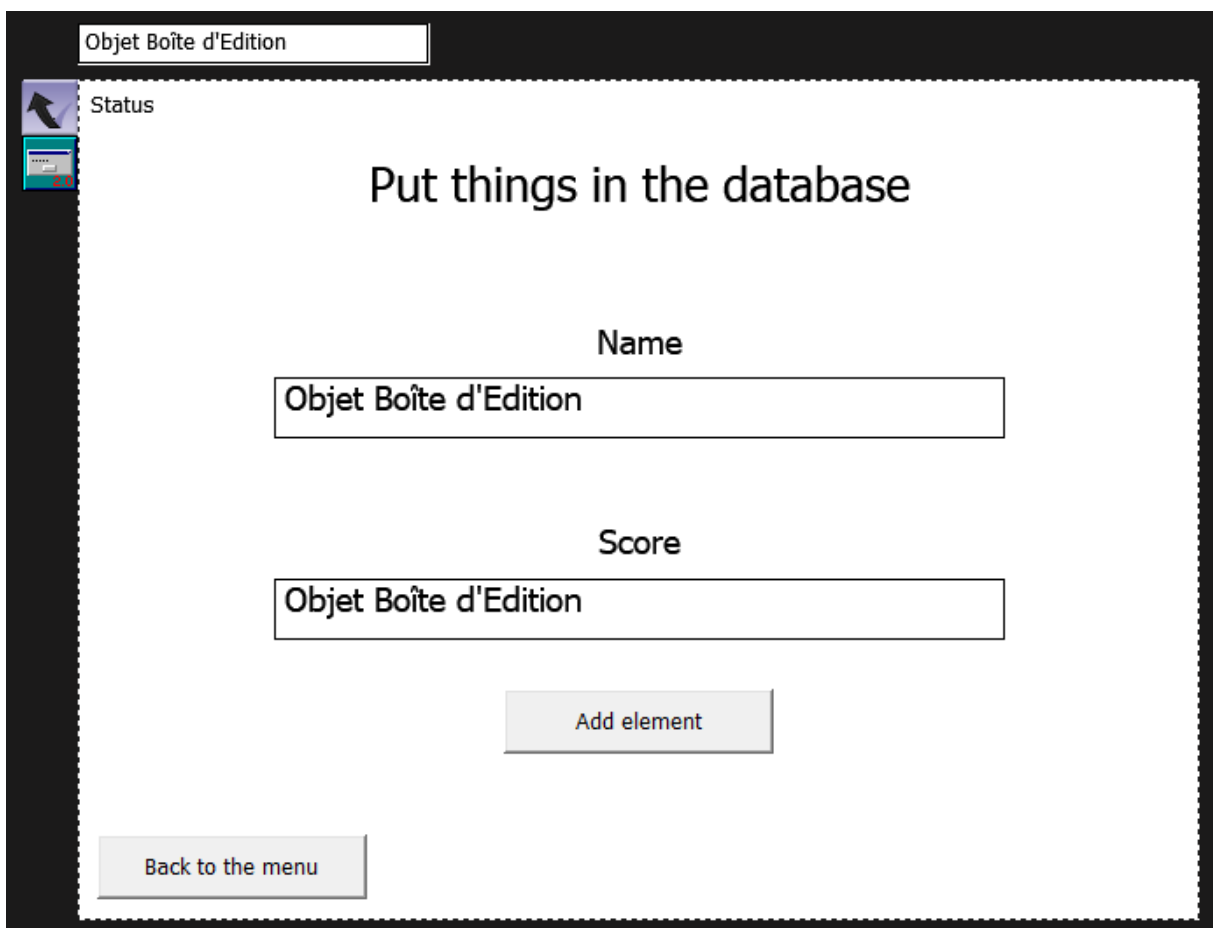
I assume that you have a PHP web server with mysql and that you put the script in the root of the server's storage and that you set up the script with your database's credentials. Now we're gonna go on Clickteam Fusion and I'm gonna explain how the mfa works, you will change it of course to your needs.

So in our application we have 2 main frames.



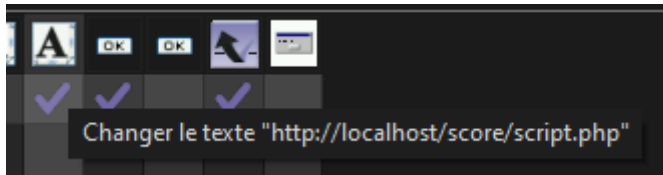
One for putting stuff in our database and another one to show them all.

The first frame is looking like that.

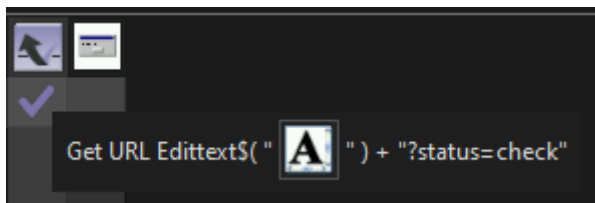
A screenshot of a software interface titled 'Objet Boîte d'Edition'. It features a 'Status' section on the left with a small icon. The main area is titled 'Put things in the database' and contains two text input fields. The first field is labeled 'Name' and the second is labeled 'Score', both containing the text 'Objet Boîte d'Edition'. Below these fields is a button labeled 'Add element'. At the bottom left, there is a button labeled 'Back to the menu'.

We have inputs texts, a button to add the element to the database and some object to do the connection to our server. Next we're gonna look into the events.

First, we have a “Start of frame” event and in it we specify the URL to our PHP script we edited before, so change that.



We disable the button to not allow the user to add element if the server's offline and in last, we do a GET request to this URL with the Get Object.

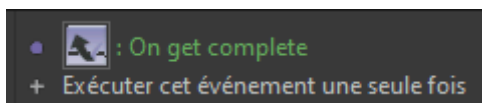


The “?status=” is a GET variable, the thing that goes after isn't important (you actually can replace the “check” text with whatever you want). The block of code in the script related to that condition is this one.

```
if(isset($_GET['status'])) {  
    echo "working";  
}
```

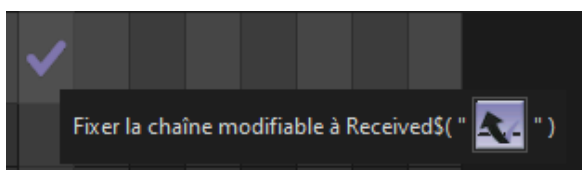
It's for verifying if the server's alive and output the text “working” that we will use in our application.

Then we do a check when the request is complete and just one time.



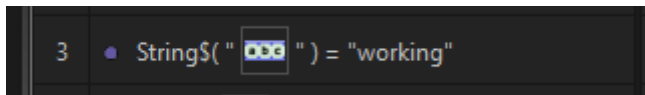
(You just could delete the “execute this event once” but that for the warning window at the end).

And we set a text on the screen to what the website is showing when doing the request.

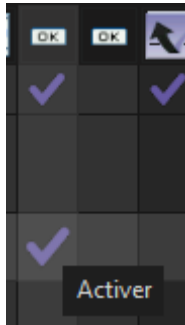


It will show our text “working”.

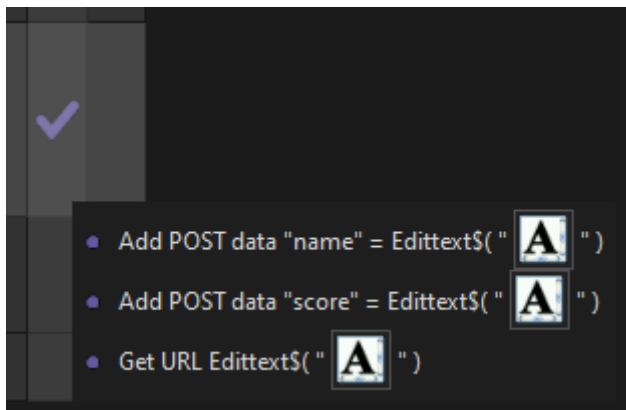
Next, when the text is equal to “working”.



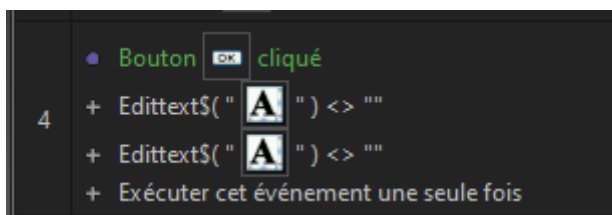
We just reactivate the button as we know the server's online.



We then create a POST request with our values.



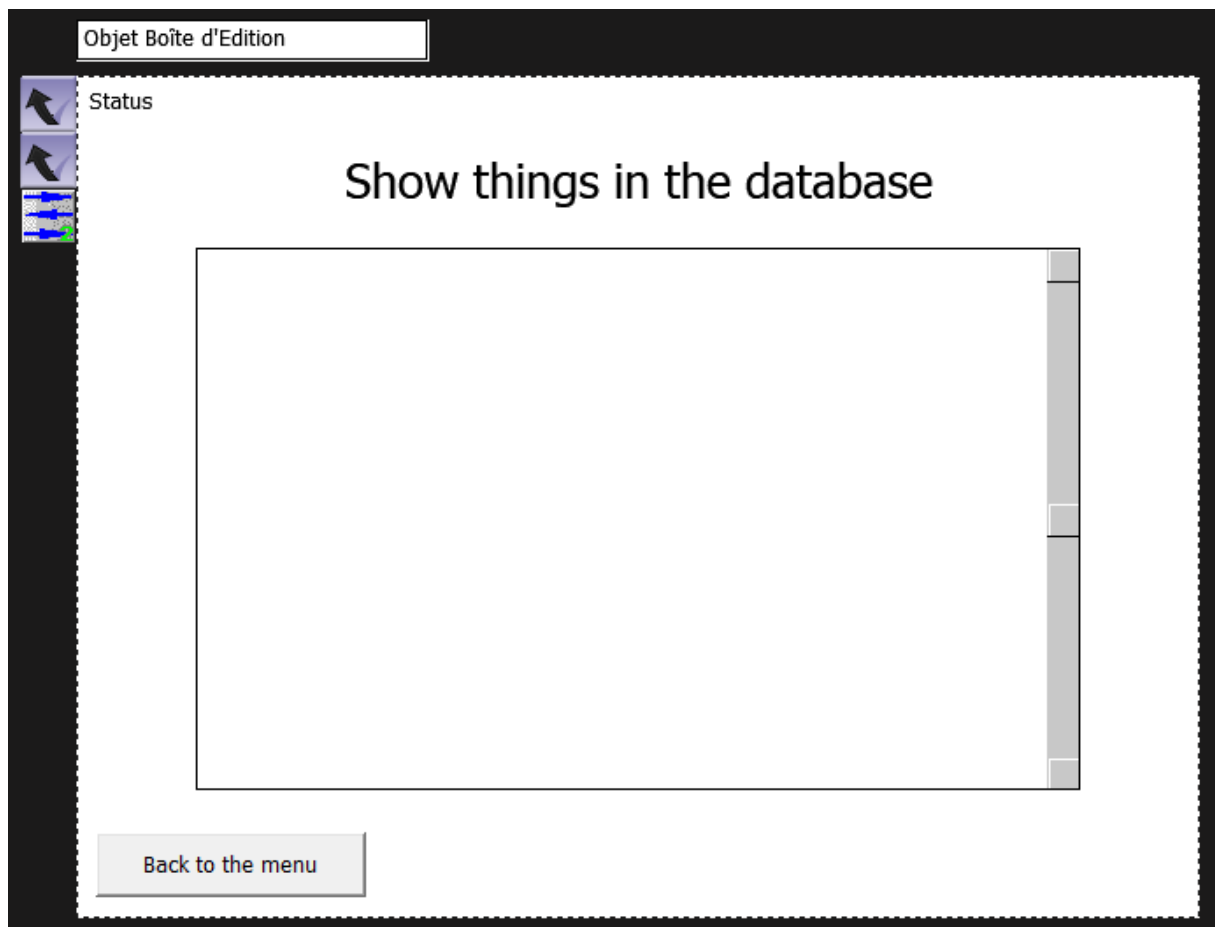
And send them to the URL we set earlier ONLY if we pressed the “add button” and that every texts inputs aren't empty and execute all of this just one time.



And if the POST request was a success, we just show a nice window.

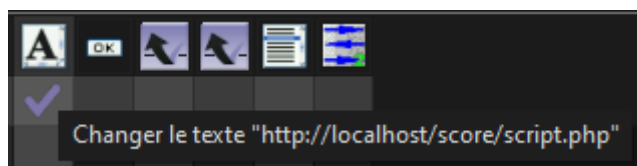


We're gonna next, look at the second main frame.



We have an input text to store the URL, a list to show all of the elements from the database and some objects to make things work. Now let's check the events.

We still have our usual "Start of frame" to store the URL of our script.

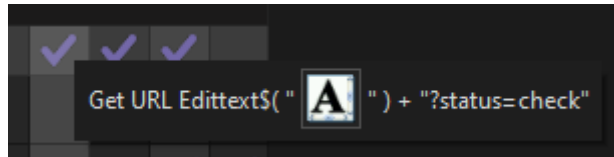


We next have a group.

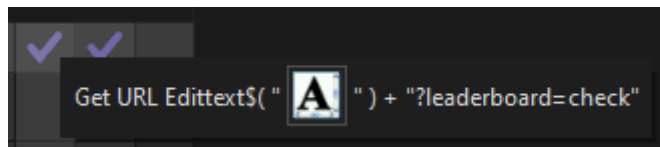
2 Check Server Status

With first a new “Start of frame” event with 3 conditions:

- One to check if the server’s alive

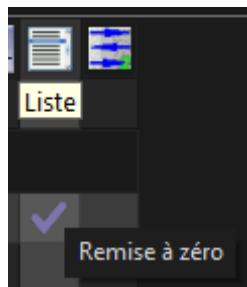


- Another one to get all of the score



(This “leaderboard” get variable is the last giant block of code in our script)

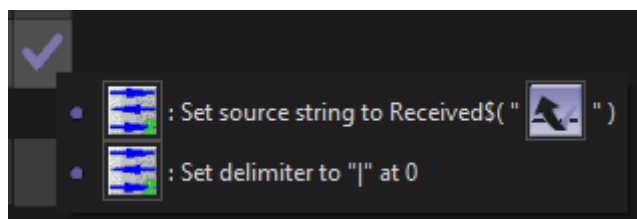
- And a last one just to reset the list



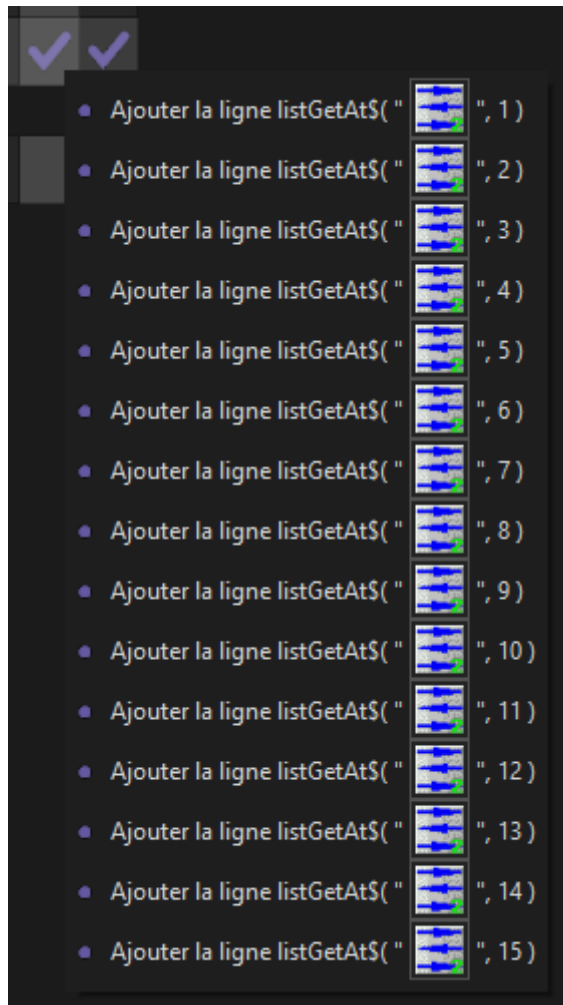
We still have the “on get complete” event to show in the screen the text “working” to show that the server’s alive. And lastly we have this.



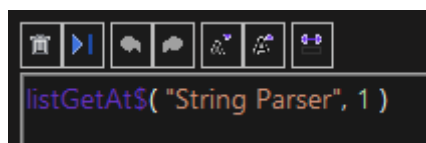
The String Parser 2 Object is here to get the response value of our second Get Object (all of the values from the table of the dabatase) then we specify the delimiter that will be used to go to a next line, the one we set in the code.



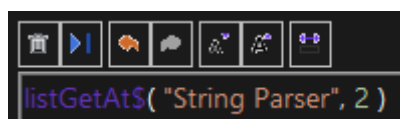
Then we add as many list as we need with on each line, the string parser's x element (where x is the index of the element).



The expression is looking like this.

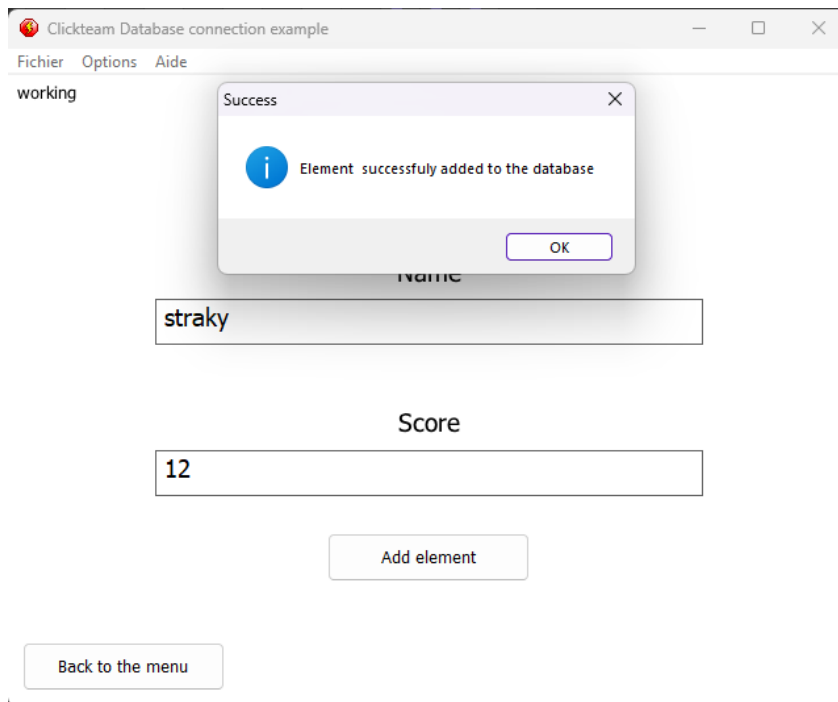


Increment the index (the 1 in the screenshot) on each “add line” condition like in the previous screen with 15 “add lines” conditions so the second conditions will be this.



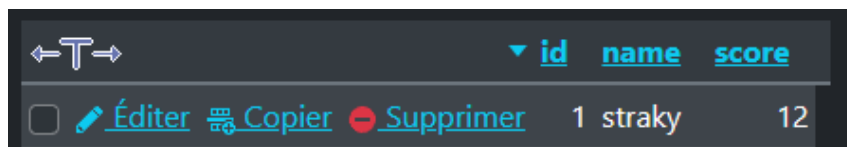
And so on.

And now the project should work, you should be able to add an element to the database using your application.



The screenshot shows a web browser window titled "Clickteam Database connection example". The browser's address bar shows "working". A modal dialog box titled "Success" is displayed, containing an information icon and the text "Element successfully added to the database", with an "OK" button. Below the dialog, there is a text input field containing "straky", a label "Score" above another text input field containing "12", and an "Add element" button. At the bottom, there is a "Back to the menu" button.

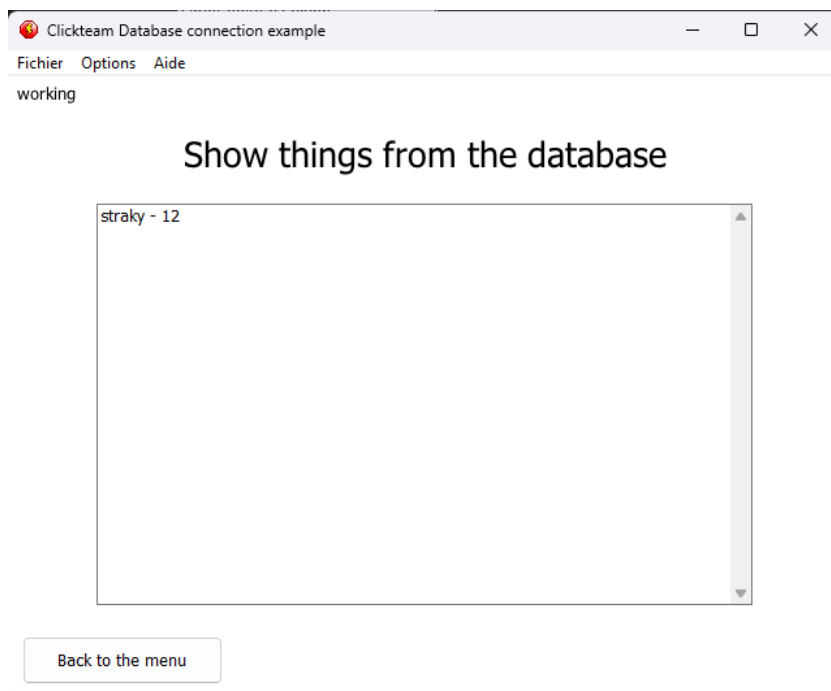
See it on the database.



The screenshot shows a database table with three columns: "id", "name", and "score". The table contains one row with the values "1", "straky", and "12". The table has a header row with the column names and a data row with the values. The table is displayed in a dark theme.

id	name	score
1	straky	12

And in our application.



The screenshot shows a web browser window titled "Clickteam Database connection example". The browser's address bar shows "working". The page displays the text "Show things from the database" above a text area containing "straky - 12". At the bottom, there is a "Back to the menu" button.