Web Databases

Contents

SQL

Setting Up the Databases

My server has a CPanel interface. You can use **MySQL Databases** to configure/add databases & users, and **phpMyAdmin** to configure/add tables for these databases. You need to use **Remote MySQL** to configure which external hosts can access your databases.
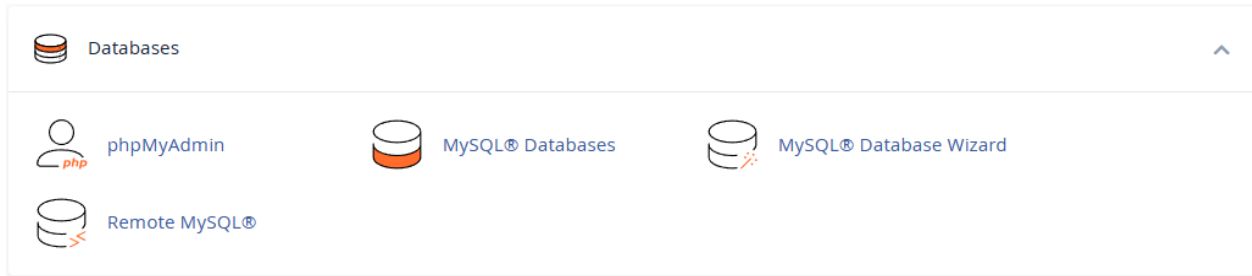
IMPORTANT NOTES:

- Foreign Key names must be unique, NOT JUST in the table, but in the entire database.

Security Issues to Overcome

I was banging my head against the wall trying to access my databases from my desktop (vs a server PHP script) and it kept failing with Error 1045 see below.

```
C:\Users\Alec\Desktop\Elegant Electrical>SQLTest.py
Error: '1045 (28000): Access denied for user 'testuser'@'ip68-108-83-8.lv.lv.cox.net' (using password: YES)'

C:\Users\Alec\Desktop\Elegant Electrical>SQLTest.py
MySQL Database connection successful

C:\Users\Alec\Desktop\Elegant Electrical>
```

After I added 68.108.83.8 (see error info above) to the list of Access Hosts in **Remote MySQL** the problem went away and I can access my databases from Python scrips on my client desktop.

The reason for this is that you don't want to leave the door open for every hacker in the world to be able to attack your database. By restricting outside access to a small list of IP addresses, this effectively closes the door to hackers trying to brute force guess your password and gain access to your data.
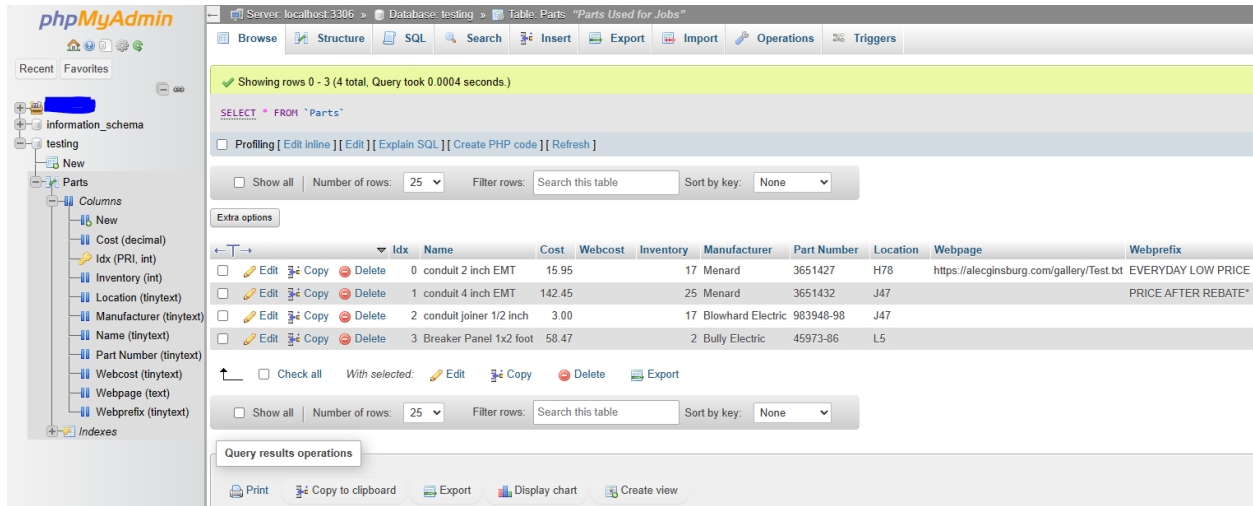
This does not prevent people "using" your web site (by browsing pages) from accessing the database using the features you programmed into the web pages. This is because these database accesses occur "within" the same computer (the server) as the database.

68.108.83.8, in this example, is my ISP temporarily assigned IP address. You can retrieve this from the connection settings of your cable modem. If you are connecting over a VPN, it will be different – it will be the IP address of your VPN server connection.

Manual Setup (Website Tools)

Although you can setup tables programmatically, it is far easier in most cases to simply use phpMyAdmin (or some similar tool) on your server admin site.

Note that you obviously only have to do this once. You can also add some test data manually using this tool which is helpful to get started.

Remote Setup (MySQL Workbench)

MySQL Workbench is a powerful (free) tool that allows you to work with any SQL database to design, implement, and document the data structure.

You can download MySQL Workbench here:

https://www.mysql.com/products/workbench/

Connecting to GoDaddy SQL databases requires you to create a test user (so you don't have to use your admin account) and enter the information shown below. Note that you have to skip through the warning message that some features may not work – this is because it is a MariaDB and not a pure MySQL database. Also the constraints outlined in the Security Issues section apply here.
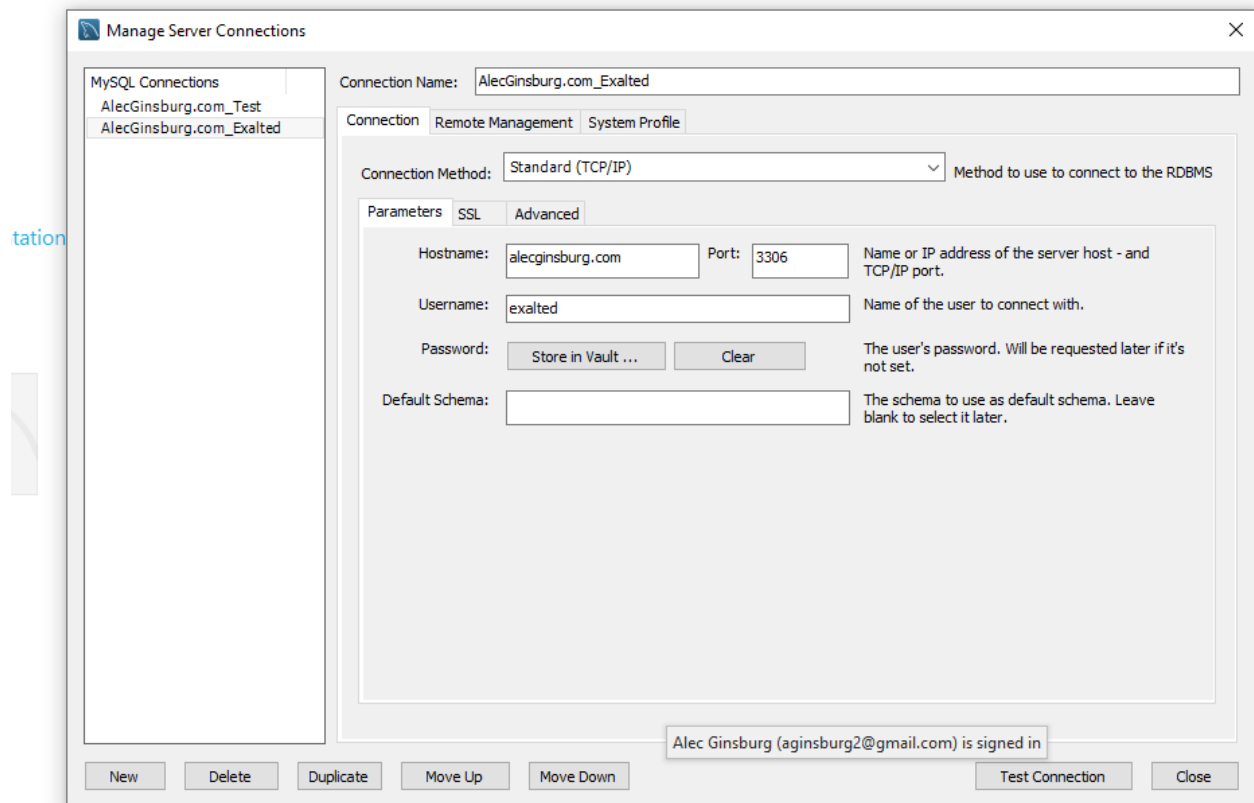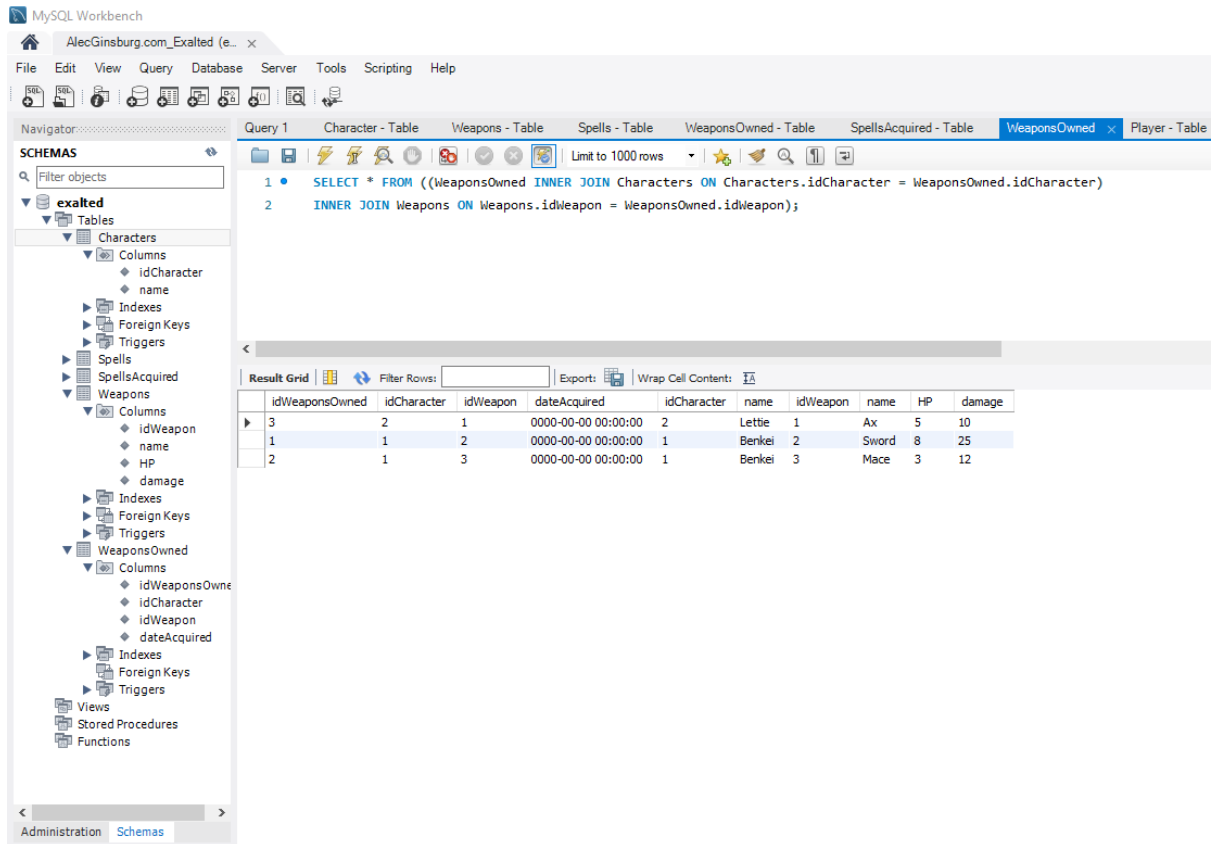
Once you have this setup, you are then free to use the tool to tables in your database, add data to these tables, modify it, create queries, etc.

The screenshot below gives you an example of some elementary tables required for a role-playing game, along with a sample query that links three tables together to show which characters "have" one or more weapons along with the player and weapon data.

Using this tool is much easier than hand editing the database on your server and allows you document the relationships to make them clearer and more maintainable.
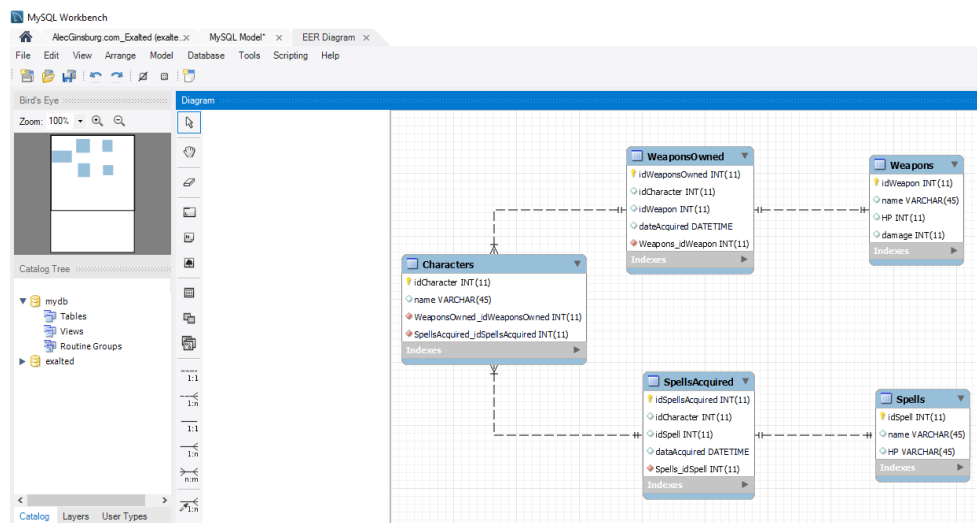
Testing SQL Syntax

If you get your SQL syntax wrong, queries can mysteriously fail and it is very difficult to figure out what went wrong. Using phpMyAdmin you can test out SQL statements to see if they work the way you expected. Here is an example of a query that "should" add a new record to the table but kept failing mysteriously.
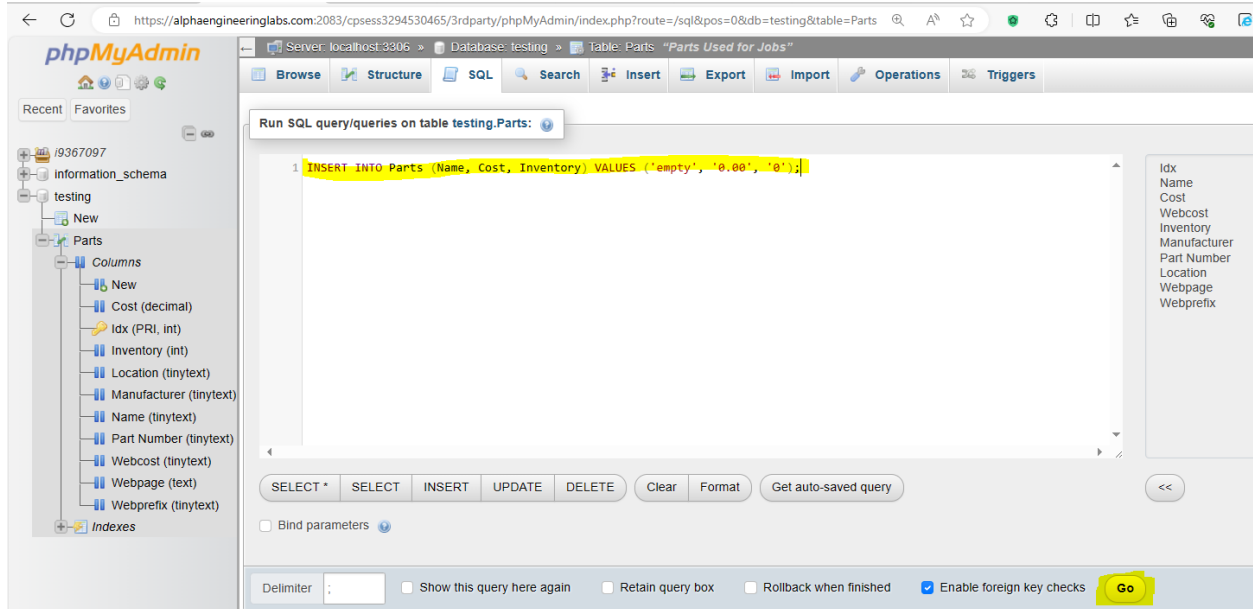
However, when it was run in phpMyAdmin the problem became immediately apparent. The 'Idx' field (which must be unique) was a duplicate of an existing record.
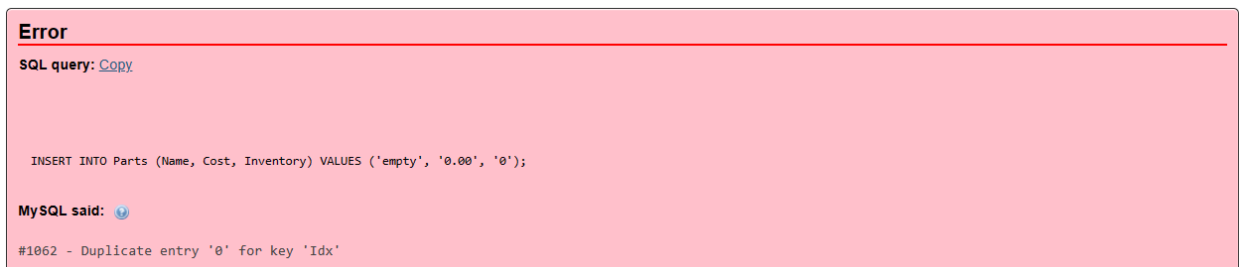
In this case the "Auto-Increment" feature was cryptically shown as "A_I" and I missed it. Activating this feature fixed the problem – BUT it was NOT as easy as just checking the box!
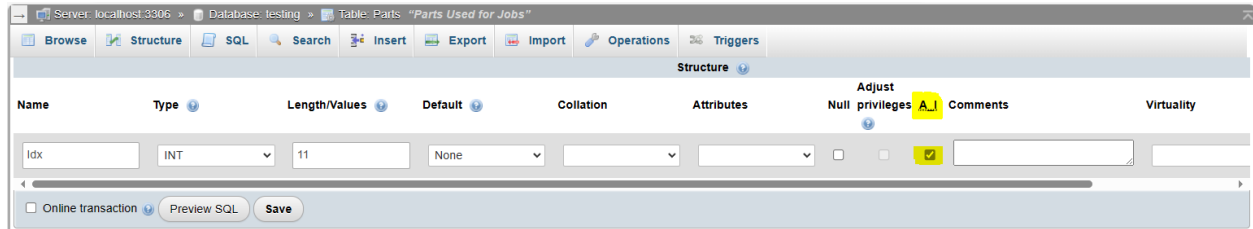
You have to check the box (or whatever other change you want), then click the "Preview SQL" button which will show the SQL required to run to make these changes.
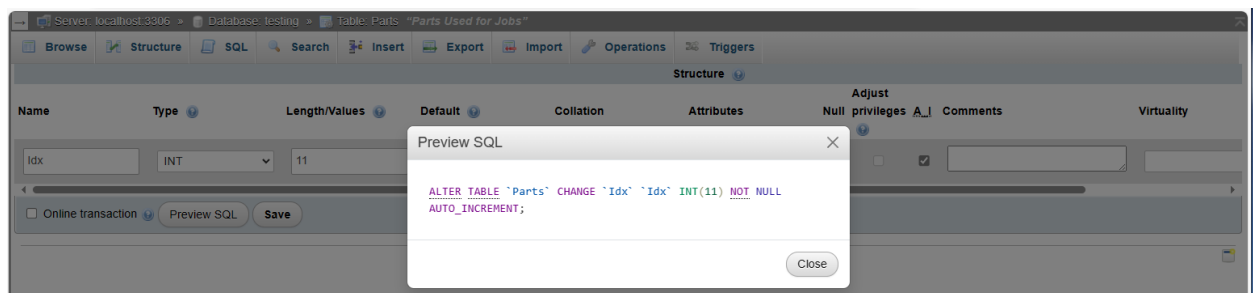
You should then paste that into the SQL tab and hit "Go" to execute it (you would think pressing the "Save" button would do it but it does NOT.)
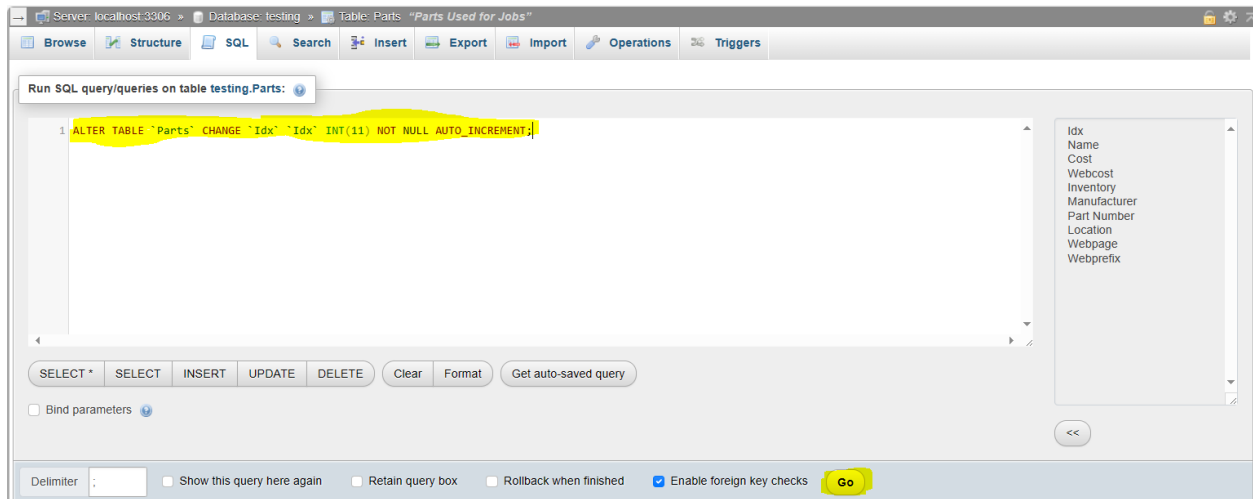
In this case, even this failed. It turned out that I had records starting at Idx=0 (which is illegal) so I had to manually change the Idx values to start at 1 before running this SQL command.

After successfully running this, the original "INSERT INTO …" SQL started to work properly.

PHP (Hypertext PreProcessor)
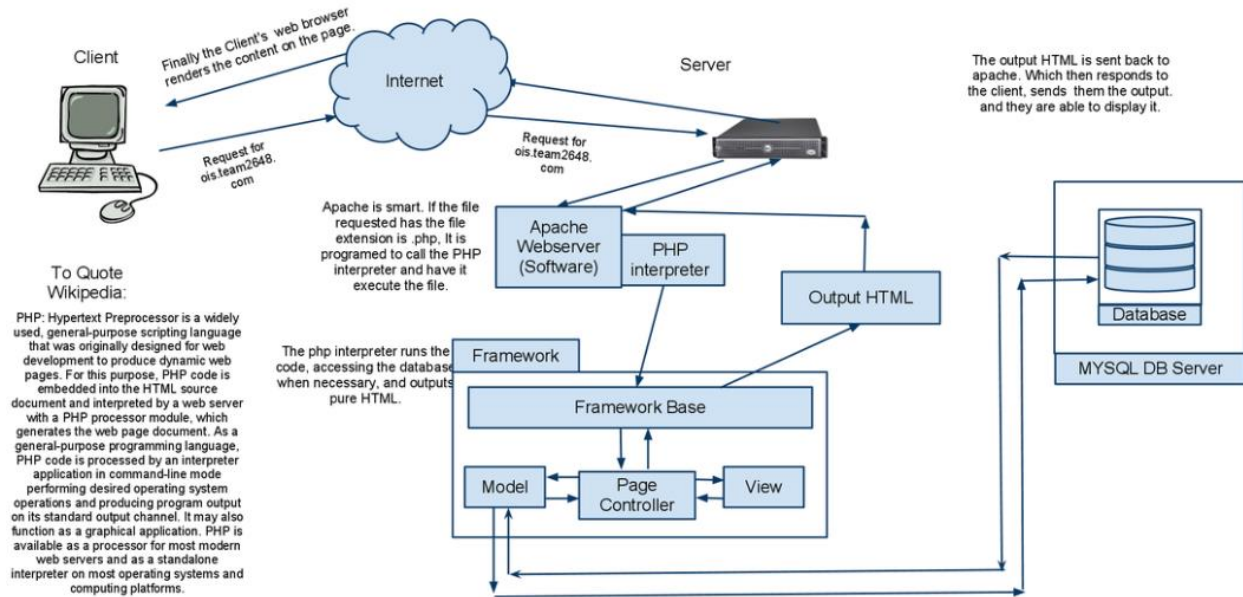
The PHP script you write and store on the server is NEVER seen by the client, instead the client sees an HTML file that is generated by the server using your PHP script.

This has many benefits:

- Usernames & passwords required to access your databases are not visible to the client.
- Database activity can be performed on the server side making the page load faster.
- The page is dynamic and can respond to changing conditions – vs a typical "static" page
- This can be used, not only to load the main page, but also to load content into a "piece" of the page (ex: a database query could be formatted and loaded into a "<div>" on the page.)

This has some "gotcha"s too:

- <script> code generated by the PHP works in the client when loading a full page, but silently fails when dynamically loading content into a <div>. This is extremely frustrating if you don't know about it. Refer to the following section for more details.
  - Technically this is an HTML issue (not PHP.)

There a few drawbacks as well:

- Database accesses have to be broken into two pieces: the request on the client script side, and the PHP server script that actually does the database work. This is a small price to pay for the security of not divulging your sites user/password.

Including/Referencing a <script> from inside Generated PHP Code

Including a <script>...</script> section in your PHP generated code **DOES NOT WORK** (see below.)

```
echo "<script>";
echo "function myfunction(){";
echo "fetch('php/db_get_testdata.php?action=updateCost&idx=1&value=567.89', { method:
      'POST' }).then(res => {console.log('Request complete! response:', res);});";
echo "}";
echo "</script>";

echo "<button onclick='myfunction()'>Change Idx 1 Cost</button>";
```

But including the script inside of the HTML elements DOES work, but is extremely cumbersome and confusing and it only works for one row entry.

```
echo "<button onclick=\"fetch('php/db_get_testdata.php?action=updateCost&idx=1&value=" .
      ($lastValue + 1) . "', { method: 'POST' }).then(res => {console.log('Request complete!
      response:', res);});\">Add $1.00 to Idx 1 Cost (refresh page to see result)</button>";
```

This also requires logic to count the rows ($rowCnt) and store the last value of 'Cost' in $lastValue which is also a burden.

It appears as though the PHP script section is either NOT parsed, or is parsed, but into a different namespace or something. This is probably due to the fact that "this" PHP is loaded into the innerHTML of a <div> on the main page and NOT into the <body> as script normally would be.

Here is an article that describes this:

https://stackoverflow.com/questions/1197575/can-scripts-be-inserted-with-innerhtml

Another possible (other than multiple <form> objects) solution: add a single pixel image with an onload callback after the script:

```
<script>
 alert("This script always runs.");
 script01 = true;
</script>
<img src =
"data:image/gif;base64,R0lGODlhAQABAIAAAAAAAP///yH5BAEAAAAALAAAAAABAAEAAAIBRAA7"
     onload="if (typeof script01==='undefined') eval(this.previousElementSibling.innerHTML)">
```

This would run the 'img onload' script which would then 'eval' the previous element (the script) which should make it available. Again, this is obtuse and confusing.

Accessing Databases from PHP

Single-Statement Queries – The Basics

Here are the basic steps to access an SQL database from PHP. Note that $offset, in this case, is the record offset that we are trying to find.

1. $servername = "localhost";
2. $username = "testuser";
3. $password = "your password here";
4. $dbname = "testing";
5. $conn = new mysqli($servername, $username, $password, $dbname);
6. if ($conn->connect_error) { die("Connection failed: " . $conn->connect_error);}
7. $sql = "SELECT * FROM Parts OFFSET " . $offset . " ROWS FETCH NEXT 1 ROWS ONLY";
8. $result = $conn->query($sql);
9. $conn->close();

- Steps 1-4 specify your database location and the valid user id that is authorized to access this database.
- Step 5 attempts to open the connection to the database.
- Step 6 aborts if the database connection failed for any reason.
- Step 7 sets up the SQL command (the subject for another chapter) to be executed.
- Step 8 executes the SQL query and returns the results. "$result" contains the actual data returned from the database as a result of your query. You will use this to finish building the reply to the client.
- Step 9 closes the connection.

You can then use stuff like this to build the client reply – this builds an HTML table row for each of the records in the database. Note that the strings in [] are the table column names in the database.

- while($row = $result->fetch_assoc()) {
- echo "   <tr>";
- echo "   <td>" . $row["Idx"] . "</td>";
- echo "   <td>" . $row["Name"] . "</td>";
- echo "   <td>" . $row["Cost"] . "</td>";
- echo "   <td>" . $row["Inventory"] . "</td>";
- echo "   <td>" . $row["Webpage"] . "</td>";
- echo "   </tr>";
- }

Multi-Statement Queries – More Advanced

Sometimes you need to guarantee the order of execution of two (or more) SQL statements. For example, when you update a record (by applying an offset to an existing value – value++) and want to then see the resulting values.

For this, it is critical that INSERT/UPDATE be 'atomically' executed with the SELECT so that the data returned by the SELECT actually reflects the result of the INSERT/UPDATE. If these two queries are submitted separately, there is no guarantee that the INSERT/UPDATE will complete before the SELECT is executed (and from experience it often is not.)

Submitting both INSERT/UPDATE and SELECT in the same query is obvious and works great in SQL Workbench, but is much more complicated using the PHP mysqli libraries.

First of all, mysqli will ONLY accept 1 SQL statement when using query(). This is for security reasons and is used to prevent SQL injection attacks. When using multiple SQL statements, you need to use multi_query() and the processing logic is much more complicated because the result of each SQL statement is returned separately so you have to iterate through them picking out the data from the pieces you care about.

Secondly, and this is due to my ignorance, I was attempting to wrap the SQL statements into a transaction using 'START TRANSACTION; ...; COMMIT;'. While this does work, it presents other problems that I did not have the patience to investigate. Specifically, this leaves the SQL engine in a state where it will not accept any other requests until 'something special' (what I don't know) happens. So, the 'next' query after this fails. I found that using transactions was not necessary for what I needed - which was simply to ensure that the INSERT/UPDATE completed BEFORE running the SELECT - so I simply removed it and the logic is working fine now.

You can see below an example of how to do this:

```
function login_failure($conn, $playerName){

    // get the IP address of the client
    $ip = $_SERVER['REMOTE_ADDR'];
    $invalidRunCount = 0;

    $sql = <<<END
      INSERT INTO IPs (ipAddress, numInvalidLogins, timeOfLastInvalidLogin, invalidRunCount)
        VALUES ('$ip', (numInvalidLogins + 1), CURRENT_TIMESTAMP, (invalidRunCount + 1))
      ON DUPLICATE KEY
        UPDATE numInvalidLogins = (numInvalidLogins + 1), timeOfLastInvalidLogin =
CURRENT_TIMESTAMP, invalidRunCount = (invalidRunCount + 1),
          blocked = CASE WHEN (invalidRunCount >= 5) THEN 1 ELSE 0 END;
      SELECT IPs.blocked, IPs.invalidRunCount FROM IPs WHERE IPs.ipAddress = '$ip';
END;
    // execute the multi-line SQL query
    $conn->multi_query($sql);
```

```php
    // loop through and check each of the return values for each of the SQL statements
    do {

        // break out of the loop if SQL sub-statement currently being processed failed
        if ($conn->errno !== 0) {
            //echo "Multi query failed: (" . $conn->errno . ") " . $conn->error;
            break;
        }

        // get the results from this SQL sub-statement
        $res = $conn->store_result();
        // both store_result & use_result will return false
        // for queries that do not return results (INSERT for example)
        if ($res !== false){

            // get the first row of data from the SELECT query
            // NB: you can also use fetch_all() to get all the rows at once and iterate through them
            //     or fetch_array() to get the rows as an array
            $row = $res->fetch_assoc();

            // pick out the SELECT data elements we care about and do the processing
            $blocked = $row['blocked'];

            if ($blocked!=0){
                // notify the appropriate entities that this IP has now been blocked
                emailDMsFromPlayerName($conn, $playerName, $ip, 'Potential Password Attack');
            }

            // free up the resources
            $res->free();
        }

        // break out of the loop if there ar no more queries to process
        if (($conn->more_results()) === false) {
            break;
        }

        // get result of the next SQL sub-statement to process
        // don't bother to check for success/failure of the result
        // since at the start of the loop there is an error check &
        // report block.
        $conn->next_result();

    } while (true);
}
```
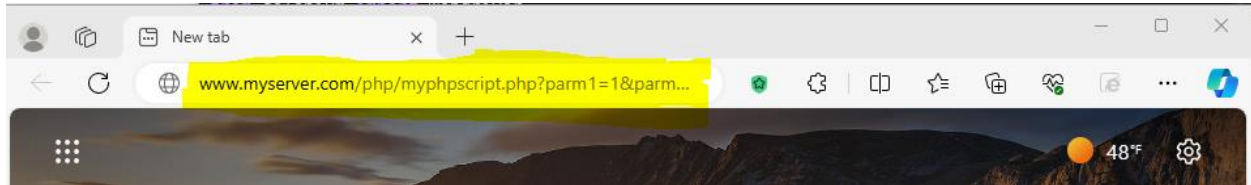
Testing your PHP Script

You don't have to get your client working to be able to test your PHP scripts. You can simply invoke them by typing in your PHP script name and parameters into your browser URL field and you will see the results in browser content window.

This is very convenient for testing.

Dealing with LONG Generated Strings

Instead of putting ALL of the PHP generated string into a single line you should use "Heredocs" to spread the information out into a readable and manageable format. Referring to the Heredocs formatted string below, imagine how difficult this would be to understand if it were all in one line!

```
echo <<<END
  <td>
    <input
      onchange='document.getElementById("sButton{$rowCnt}").style.backgroundColor="yellow";'
      onkeypress="this.onchange();"
      onpaste="this.onchange();"
      oninput="this.onchange();"
      type="number"
      id="inventory"
      name="Inventory"
      value="{$row['Inventory']}"
      form="table_form{$rowCnt}"
    >
  </td>
END;
```

Note how variable are expanded inside double quotes (without the usual "text" . $rowCnt . " more text".) It is important to enclose the variable in {} to prevent the parser from gobbling extra characters past the end of the variable name.

IMPORTANT – newer versions (PHP 7.3.0 and later) of PHP allow the "END;" to be indented. My version which shows as 7.4 still does not support this, but after upgrading to 8.1 this started working.

IMPORTANT – some web editors do not consistently color code the "<<<END" and "END;" blocks. This is annoying, but as long as you don't see the red 'X' control in your script you are probably OK.

JavaScript

JavaScript runs on the client side to respond to client input and request data from the server.

URL Parameters (or how to pass data TO the server)

PHP works great to format data being sent from the server to the client, but how do you get data to go the other way – from the client to the server?

You do this in JavaScript by appending the data to the end of the PHP URL. For example, your client code might be trying to run the server "db_get_testdata.php" script with a "fetch" JavaScript command as follows:

```
fetch("php/db_get_testdata.php ", { method: "POST" })
```

But you might want to only get the second record (and not all of them) so you need a way to be able to specify this. This is accomplished by appending the request parameters to the end of the URL as follows:

```
fetch("php/db_get_testdata.php?action=getNth&offset=n", { method: "POST" })
```

In JavaScript this process is automated for record editing using the <Form> control (see Using <Form>.)

Then in your PHP script you can extract the request parameters using the following code:

- $url = $_SERVER['REQUEST_URI'];   // get full URL passed
- $url_components = parse_url($url);   // parse it
- parse_str($url_components['query'], $params);   // pick out pieces after '?' into $params

Then you can check the parameters as follows and change the PHP logic flow based on the parameters.

```
if($params['action']=="getNth") …
```

Using <Form> to Update DataBase Records

Forms are a handy way to group together all the data related to a SINGLE database record so that the data can be sent to the server in one piece (see URL Parameters.)

Using a <Form> allows you to tie together all of the record data AND provides the logic to "Submit" the data to the server – thus appending the data as parameters to the Post request and handling all of the processing logic for you.

In general, you should prefer to use a <Form> for record updates, than to hard coding the logic yourself.

Using <Form> Inside of a <Table>

There are many times when you will need to use a <Form> inside of a <Table> - when you want to be able to edit many records at once. An example is shown below.

Current Database Values

name	cost	inventory	webpage	
conduit 2 inch EMT	16.95	18	https://alecginsburg.com/gallery/Test.txt	Submit
conduit 4 inch EMT	3.95	1925		Submit
conduit joiner 1/2 inch	4.25	15	www.test.org	Submit
Breaker Panel 1x2 foot	60.47	3		Submit

Unfortunately, you can't put a <Form> inside of a <Table>, however there is a work-around that allows you to define the <Form> outside the <Table> and 'reference' it from inside the <Table>. This logic can be seen on the following pages with comments describing the operation.

Note that this logic is actually PHP code, but illustrates the JavaScript limitations, and how to work around them.

Using this logic, the user can change any of the editable fields in any of the records displayed and the press the "Submit" button to update the changes for that record back into the server database.

Note that $queryResult is the results of your SQL query as follows:

```
$conn = new mysqli($servername, $username, $password, $dbname);
$queryResult = $conn->query("your SQL here…");
```

This logic gets the table style sheet, creates a <Form> for each data record to display (note the highlighted form id that is used later to reference this form), then creates the <Table> and its header section.

```
echo "<link href=\"https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css\"
rel=\"stylesheet\"/>";

// Create a separate form for each row, each having a different 'id'
$rowCnt = 0;
while($rowCnt < $queryResult->num_rows) {
   echo "<form action='php/db_get_testdata.php' method='get' id=table_form" . $rowCnt .
           "></form>";
   $rowCnt += 1;
}

// Create the table with appropriate formatting & column headers
echo "<table id='testTable' class='table table-striped table-earning'>";
echo "<thead>";
echo "   <tr>";
echo "      <th>name</th>";
echo "      <th>cost</th>";
echo "      <th>inventory</th>";
echo "      <th>webpage</th>";

// putting 'hidden' fields on the right side removes the 'gap' left on the left side and looks better
// don't show 'hidden' field column headers
echo "      <th></th>";
echo "   </tr>";
echo "</thead>";
```

This logic then creates the <Table> body and populates it with a separate <tr> entry for each data record to display. Note how each data entry field (and the 'Submit' button) that may be updated to the server has a highlight form=table_form9 that "ties" it back to the appropriate form defined above. Additionally, the 'Idx' record column is 'hidden'. This is required because it needs to be sent to the server but we don't want the user to see it or change it.

```php
// Create the table 'body' by generating a <tr> row for each entry in the SQL query result
echo "<tbody id='testBody'>";
$rowCnt = 0;

while($row = $queryResult->fetch_assoc()) {
   echo "   <tr>";

   // Columns that the user can't/shouldn't change are displayed as simple text
   echo "      <td>" . $row["Name"] . "</td>";

   // Columns that the user CAN change are shown as <input> fields associated with
   // the correct form (form=) and column name (name=)
   echo "      <td><input type='number' id='cost' name='Cost' value='" . $row["Cost"] . "'
            form=table_form" . $rowCnt . "></td>";
   echo "      <td><input type='number' id='inventory' name='Inventory' value='" .
            $row["Inventory"] . "' form=table_form" . $rowCnt . "></td>";

   // empty text strings screw up the HTML, so only show the 'value' if the text field is NOT
            empty
   // instead of: "value='' form=table_form..." you get: "value form_form..." which causes row
            columns including this, and after, to NOT be sent to the server
   echo "      <td><input type='text' id='webpage' name='Webpage' size=50" .
            ((isset($row["Webpage"]) and $row["Webpage"]!="") ? (" value='" .
            $row["Webpage"] . "'") : ("")) . " form=table_form" . $rowCnt . "></td>";

   // Hiding the 'Idx' column allows it to be included when the 'submit' button is pressed for this
            row
   // thus enabling the server PHP logic to know which record to update, but doesn't allow the
            user to see/change the value.
   echo "      <td><input hidden size=0 type='number' id='idx' name='Idx' value='" . $row["Idx"] .
            "' form=table_form" . $rowCnt . "></td>";

   // add the "Submit" button at the end of the row - disable validation
   echo "      <td><input type='submit' formnovalidate='formnovalidate' value='Submit'
            form=table_form" . $rowCnt . "></td>";
   echo "   </tr>";

   $rowCnt += 1;
}

echo "</tbody>";
echo "</table>";
```

Python

Python is a cool scripting language that can be used for many things. I used it to scrape data values from other web sites and store the results in a database.

It can be downloaded [here](#). After installation, make sure the 'bin' folder is in your path and the '.py' extension is properly associated, then you can run '.py' files by typing the file name in a command prompt.

Modules

Python supports lots of cool modules that 'extend' the language. You can install them from a command line as follows:

- python -m pip install selenium
- python -m pip install bs4   (this is pandas)
- python -m pip install lxml
- python -m pip install mysql-connector-python

The first 3 install modules that are handy for web page manipulation.

The 4th installs the modules necessary to access a remote SQL database.

Screen Scraping (or how to get current store pricing data)

Screen scraping is basically having your computer navigate to a website (just like you would do manually) and then examine the data presented on the screen to extract the information you care about. For example, you might want to go to a vendor's website to see the current price of some product that changes over time.

There is nothing illegal about screen scraping as long as your program does it "as a human would." If a human can press a button every second, then it's ok for your program to programmatically press the same button once a second. But it NOT ok for your program to press the button a thousand times a second as it can interfere with the website operation. Therefore, you must make sure that there are ample delays in the right places in your code.

Here is a novel way to get data by "watching" network traffic: weblink.

Here is the way I chose to do it: weblink. This uses a Python Selenium module to "take control" of your web browser (Chrome) which is pretty cool. See Example Code further down in the document.

CAPTCHA Issues

Most websites frown upon screen scraping and discourage its use by presenting CAPTCHA blockers that require the user to prove they are human by solving some puzzle. This is probably to prevent their competitors (not their customers) from automatically getting their prices and then adjusting all of their own prices to be lower so they can steal the stores customers.

The CAPTCHA is triggered by some observed client behavior on the server – and it may be different on each website. However, if the CAPTCHA check is passed successfully, the server will generally allow further activity.

So, in my case where I just need to get updated pricing once a week or so, I just manually run a Python script that gathers the data and solve the CAPTCHA challenge myself. This simply requires that I allow more time (say 20 seconds) in the script for the first interaction – so that I have enough time to solve the CAPTCHA, and then subsequent automated accesses can proceed at normal speed (say 3 seconds.)

Blocking Issues

I ran into a very annoying issue when I started trying to have my Python script be able to actually use the web page "search" feature (so that I could go to the page for a specific part number.) I have no idea what triggered it – undoubtably something I did wrong – but the site started displaying the following message every time I tried to perform a search:

> **We're sorry, an error occurred during the search for "3651427". Please wait several seconds and try again.**

I spent about a full day trying to figure this out.

- Initially I thought it was something in my Python script, so I tried taking everything out (except the page open) and it still blocks the search.
- But if I open the same page manually in Edge (a different browser) I can successfully search for the same item that was blocked in Chrome.
- Trying to give the website a 15-minute timeout to see if it starts working again. It still fails and requires the CAPTCHA check
- ALL OF THE FOLLOWING TESTS ARE "MANUAL" - NO PYTHON SCRIPTS, JUST USING THE SITE AS A NORMAL PERSON WOULD
  - Waited another hour and tried to search for 3651427 again BUT this time I MANUALLY opened Chrome to site's home page (which required a CAPTCHA check) and MANUALLY searched for 3651427 - and I still got the error. This confirms that the site has blocked Chrome on my endpoint and now it has nothing to do with the automated logic.
  - Tried waiting overnight and clearing ALL Chrome cached data (Advanced mode) and still the error occurs when searching.
  - Tried 'ipconfig /flushdns' 'netsh winsock reset' followed by cable modem power cycle and the error still occurs when searching.
  - Tried re-installing (un-install then install) Chrome and still the error occurs when searching.
  - I setup a VPN (Express VPN) routing through LA and **NOW the search feature started working!**

So, it would appear that at least some websites "detect" improper behavior and "block" that "IP/browser" configuration pair.

So, you can get around this by changing your browser, or your IP address (using VPN.)

Example Python Script

Here is some example code. You will need to tailor the 'soup.find' lines to work for the pages you are interested in processing:

```python
# Import libraries
from selenium import webdriver
from bs4 import BeautifulSoup
import pandas as pd
import sys
import os
import time
```

```python
# set delay between 1st and 2nd page to be longer so you can do the robot check if necessary
SleepSeconds = 20
# initialize the output array to be empty
OutputArray = []

# Init browser
driver = webdriver.Chrome()
driver.implicitly_wait(3)
driver.delete_all_cookies()

for page in PagesToCheck:
    # Navigate to page
    driver.get(page)
    print("WEBPAGE: ", page)
    if "Pardon Our Interruption" in driver.page_source:
        print("Page is Blocked")
    else:
        # Parse page HTML
        soup = BeautifulSoup(driver.page_source, 'lxml')
        # Identify elements to scrape
        product_names = soup.find('h1', {"data-at-id": "itemTitle"})
        product_prices = soup.find('td', {"data-at-id": "full-price-discount-edlp"})
        # Extract html data found into arrays and clean up the formatting
        names = [p.text for p in product_names]
        prices = [price.text.strip() for price in product_prices]
        # Store in DataFrame
        if len(names) == len(prices):
            OutputArray.append(names)
            OutputArray.append(prices)
            #print(names, " for ", prices)
            #df = pd.DataFrame({'Product': names, 'Price': prices})
            # Export to CSV
            #df.to_csv('products.csv', index=False)
        else:
            print("len(names)=", len(names), ", len(prices)=", len(prices))
    #sleep so the host doesn't think we are a bot
    time.sleep(SleepSeconds)
    #now go faster for the other pages which shouldn't have any checks
    SleepSeconds = 3
```