|  |
| --- |
| University of South Wales |
| Advanced Internet and Mobile Computing |
| IS3S664 |

|  |
| --- |
| Baber M (FCES)  1-10-2020 |

Table of Contents

[1 – Introduction 2](#_Toc29567127)

[2 – Design 3](#_Toc29567128)

[3 – JavaScript / jQuery 5](#_Toc29567129)

[3.1 – Introduction 5](#_Toc29567130)

[3.2 – Test jQuery 5](#_Toc29567131)

[3.3 – Dropdown 5](#_Toc29567132)

[3.4 – Get XMLS (Pets and Guitars) 5](#_Toc29567133)

[3.5 – Get PHP (Pets and Guitars) 7](#_Toc29567134)

[3.6 – Insert XML (Pets and Guitars) 8](#_Toc29567135)

[3.7 – Insert PHP (Pets and Guitars) 8](#_Toc29567136)

[3.8 – Forms 9](#_Toc29567137)

[3.8 – Search 9](#_Toc29567138)

[4 - PDO 10](#_Toc29567139)

[5 – Testing 10](#_Toc29567140)

[6 – Conclusion 11](#_Toc29567141)

[7 - References 11](#_Toc29567142)

[8 - Software 12](#_Toc29567143)

[9 – Appendix 13](#_Toc29567144)

[9.1 – Index.html 13](#_Toc29567145)

[9.2 – CSS 23](#_Toc29567146)

[9.3 – JavaScript 26](#_Toc29567147)

[9.4 – php-get.php 35](#_Toc29567148)

[9.5 – php-insert.php 37](#_Toc29567149)

[9.6 – xml-get.php 39](#_Toc29567150)

[9.7 – xml-insert.php 41](#_Toc29567151)

[9.8 – XML 42](#_Toc29567152)

[9.8.1 – Guitars 42](#_Toc29567153)

[9.8.2 – Pets 42](#_Toc29567154)

[9.9 – Database Backup 44](#_Toc29567155)

# 1 – Introduction

[Link to Website](https://ces-web2.southwales.ac.uk/students/17076749/java/index.html)

This report will look over creating a website which makes use of front-end and back-end functions using JavaScript, jQuery and PHP to manipulate the page layout. Starting with the design, the implementation and how the functions are navigating the DOM tree to send and receive data.

# 2 – Design

Before anyone can create a website, it would be ideal if they had an idea of what they want it to look like. This would have website layout and basic functions with a brief explanation of what functions they should carry out (e.g. print hello on click).

The design of this website was intended to be as simple as possible to not distract from the main operations of the JavaScript. Below was the original design which was drawn up with a little bit of information about what each button is doing and how it is displayed.

A close up of a piece of paper

Description automatically generated

After looking at the design, the next stage was to start the implementation of HTML & CSS design.

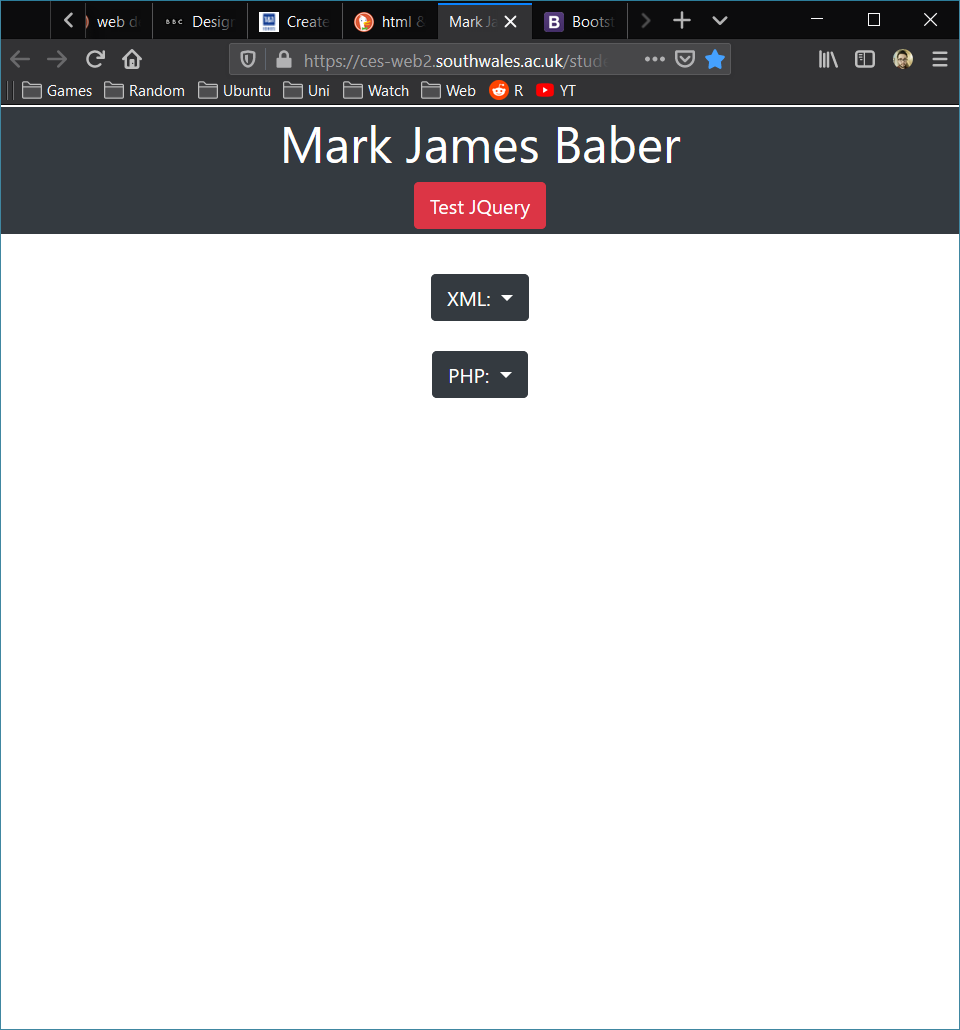
To start this process there was a frame-work used called Bootstrap which will take care of a lot of the CSS design, which can be used to speed up the process of creating a new website. This was used due to the fact the designer had used it for a few other websites in the past.

Starting with the default page and 3 buttons was quite simple without functionality, back to the design each section is within a div element to make sure we can edit each part separately if needed. For example, if we wanted to replicate the main header from the landing page, this is all the code you would need.

*<div container>*

*<h1>Mark James Baber</h1>*

*</div>*



(Landing Page)

From the landing page, the user has a choice of 2(3) buttons to choose from, the XML and PHP buttons were implemented to get data from their tables (XML & PHP). To get these buttons working correctly, there was some JavaScript & jQuery being used to create a drop-down effect. The user would then be prompted with the 2 types of files available to them. This will be covered more in the next section.

# 3 – JavaScript / jQuery

## 3.1 – Introduction

Within this section of the report, the development of the javascript and use of jQuery will be explained. Going through each of the functions and how they were implemented to carry out certain tasks.

## 3.2 – Test jQuery

During the testing phase, there was a lot of errors being made with how to import the jQuery library. Due to this, there has been a button added called *Test jQuery* at the top of the page. This was implemented to alert the user is anything went wrong with the jQuery library. This simple function works something like this:

*onPageLoad – if jQuery library detected – alert (Library loaded fine)*

*else - alert (Something went wrong).*

With 2 alerts being called with 1 function, it would get annoying every time the page loaded to alert the user, which is why it is now waiting for the click of a button.

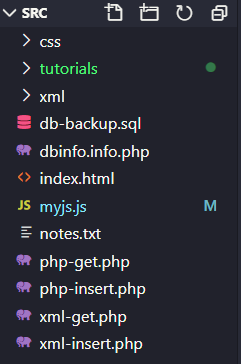
## 3.3 – Dropdown

The drop-down buttons also have JavaScript functionality, to start with both buttons have the same effects. When both buttons are click, their contents will start hidden with a delay and a slide down to make the button more modern. This can be seen quite easily within the jQuery code as some parts is quite easy to read. Below is a snippet for the first button.

*$('#dropdown-xml').hide().delay(100).slideDown(250);*

## 3.4 – Get XMLS (Pets and Guitars)

For the first option on the drop-down after the fancy effect, is the *XML Pets* button, as the name states it will get the pets from the xml file. In order to do this the file structure should be as follows

By having the xml folder contain the xml files, it doesn’t allow any errors from the server. As some servers have rules set up to make sure files can’t be manipulated at the root file.

So, in order to make sure the button XML – Pets gets the pets, there was JavaScript and PHP being implemented. To start off the function it uses a method called getJSON which calls a php file *xml-get.php.* This PHP file has a lot of information in there for error checking, so the user can track errors and easily report them. The file also contains what the source file should be called, which is stored as a variable in the PHP, which checks that a file exists within a folder called *"xml/".$filename.".xml"*. By using the filename within a variable, this allows the use of the same code for multiple files and tables.

After some extensive error checking, if no errors are found the file is returned to the JavaScript function, which now states what the file name is *sourceName: "pets"*. Again, this is great so there isn’t much to be changed for other files and tables. After getting the xml file back, the function will then go through another form of error checking to make sure the data is in fact an XML and doesn’t have any syntax errors. This was where the function started to become quite sophisticated, the function starts off by declaring a variable called table which contains the html for a table header and the first row. For this example, it was for pets, so the row looks like this:

*<thead><tr><th scope="col">Pet Name</th><th scope="col">Pet Type</th><th scope="col">Pet Breed</th><th scope="col">Pet Owner</th></tr></thead><tbody>';*

The function then checks the entries of the XML using the forEach function, this goes through the XML for all pets with the data entries within each pet. For example,

*Pets: pet1 – pet1.name – pet1.type – pet1.breed – pet1.breed – pet1.owner*

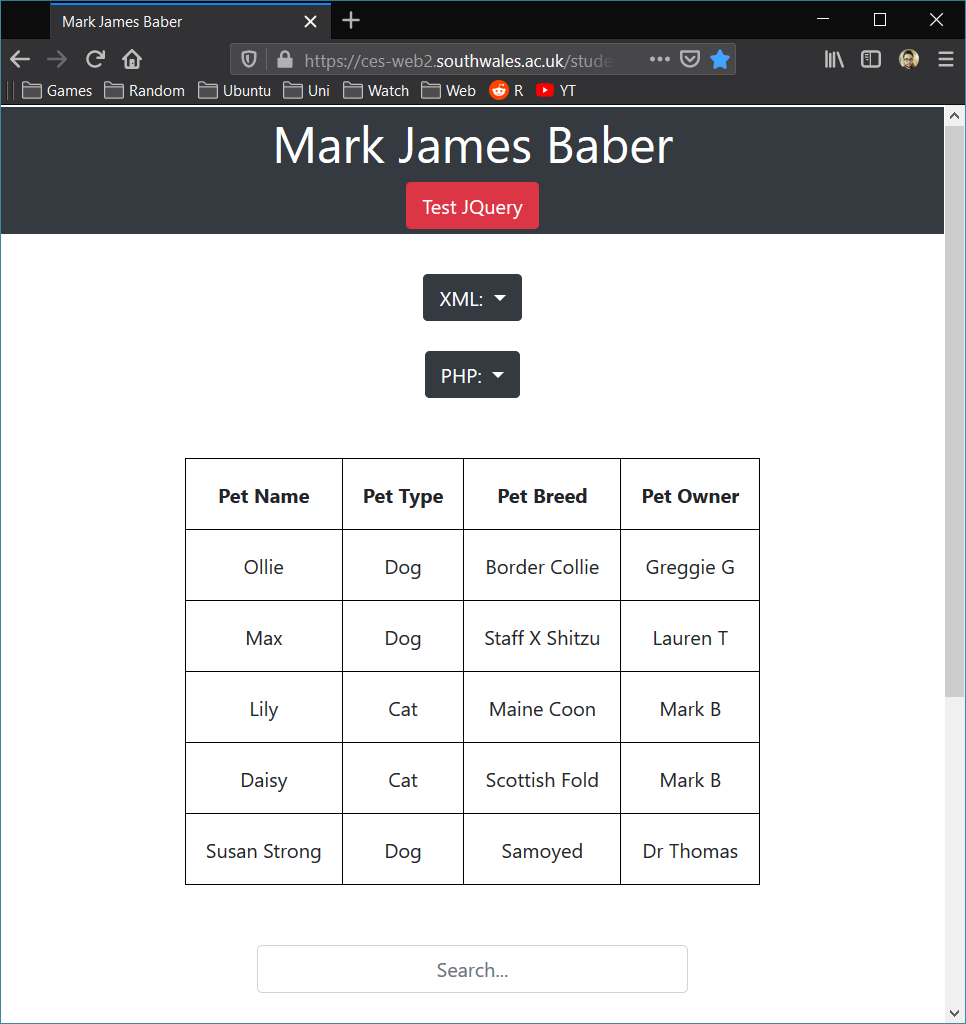
In order to carry out this function smoothly, there needed to be a nested loop getting the entries (pets) and values (petName). By putting this in a nested loop the function can carry out a loop of pets with the respected values and due to this being within a loop, it builds the rest of the table around each entry to make it dynamic. This data is then placed within a div on the index.html page using a small bit of jQuery,

*$('#dataTable').html(table);*

This function was re-used for *XML Guitars*, except for the omission of pets for guitars after the sourceName, which is how the same function can be used for multiple files. There was also a few values to be changed within the table variable for the table headings and what values to get from each array within the nested loop. Similar to the pets, for guitars it was as follows;

*Guitars: guitar1 – guitar1.brand - guitar1.model - guitar1.type*

This concludes both Get XML functions.



(XML – PETS)

## 3.5 – Get PHP (Pets and Guitars)

The next function used was to get the files from a remote server, with the use of PHP. Whilst briefly touching on PHP to set up the PDO, this function operates similarly and yet can fetch data from a server. The function starts the same as the *Get XML* which uses a method which called getJSON again, except by calling a different file which is called *php-get.php*. This file is different to the XML version as for the user to connect to a database, you would need to have some form of credentials.

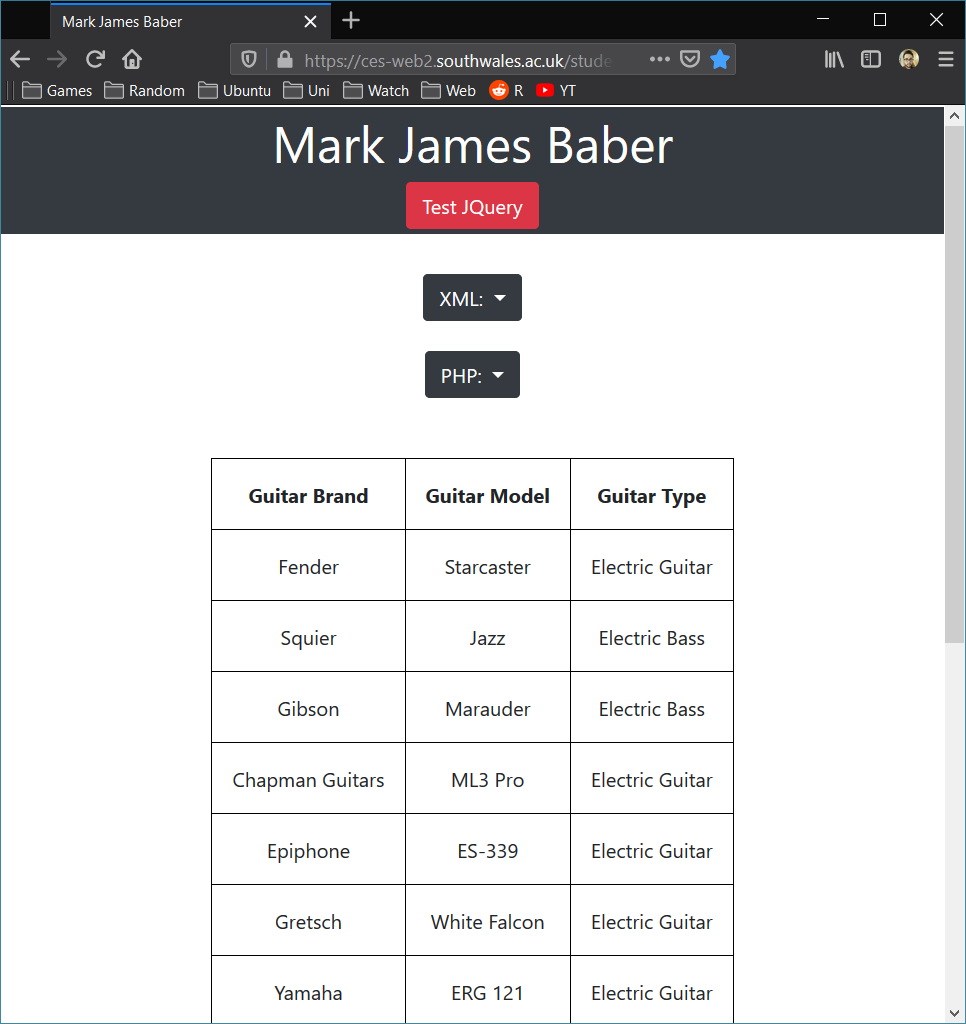
If the PHP file doesn’t find any basic errors to start with (syntax), it will start by including a file which is on the server called dbinfo.info.php. To do this in the PHP file you simply add;

*include "dbinfo.info.php";*

After including the credentials, the PHP file will then test the connection using the host, database, username and password from the dbinfo file. This is a good way to ensure you’re not passing in your credentials manually for a few reason, 1 it would be harder to get access to, especially if people don’t know your file system layout and 2, it leaves less room for human error and if things change, you only have to change it in one place.

If the PHP connection testing phase is over and returns no errors, it then starts to build up the PDO. Starting with the tableName as a variable, which it calls from the rest of the original function and then prepares the SQL before executing. The function will look to see if the database actually returns anything, or if not, return a few errors. Ranging from no table, no data within the table, or no connection to the database. After passing these error checking steps, the PHP file will then return the data to the original function.

Back to the original function, under getJSON this is where you can say what table to get the data from. This data is then passed through another function as data and will loop through again (like the rest of *Get XML*). Building the table as a variable and sending the data through a nested loop to get all the keys and values from the data, whilst building the rest of the table around each entry dynamically. After both loops have fetched all data from the database, this data and table is then placed into a div within the html called *dataTable*.



(PHP – Guitars)

## 3.6 – Insert XML (Pets and Guitars)

The next step after getting the data to display, was to make it so a user was able to input their own data. The next function here was to insert pets into the *Pets XML.* To start off with the function, the first thing to do was to create a variable which would be able to hold our information. Because the variable is expecting an input, the user is able to get the value from a text box using jQuery and assigning it to a specific variable. For example,

*var formData = new Object();*

*formData.PETNAME = $('#xmlPetName').val();*

This piece of code it is essentially saying, whatever value is located within the div with an id of xmlPetName, assign that to the formData object PETNAME. As this value is an object, the next line in the function is crucial which is *JSON.stringify(formData)* which converts that object into a string. The next step is to say what the source file is, where you want to send the data, for this example it is *pets,* which then calls another method which has been used before, getJSON.

This method calls in the *xml-insert.php* file which again checks the source file, with the specified filename and checks it is there. After doing so, the php file starts to get the entries within the file and adds a new child. By doing so, it can safely build up the *DOMDocument* to make sure when data is inserted, it goes after the last child within the source file. If there are no errors, the PHP file will return a true back to the getJSON method to carry on to the next steps.

Next step is for the *sourceName* of the file to be passed through, and if it was a success, there is a little bit of jQuery to create a small box which states “Record inserted correctly”. Which would have successfully inserted a new record into the XML file and can work for both Pets and Guitars.

Like the *Get XML* function, this can easily be used for the Guitar XML by omitting pets under the sourceFile and putting guitars.

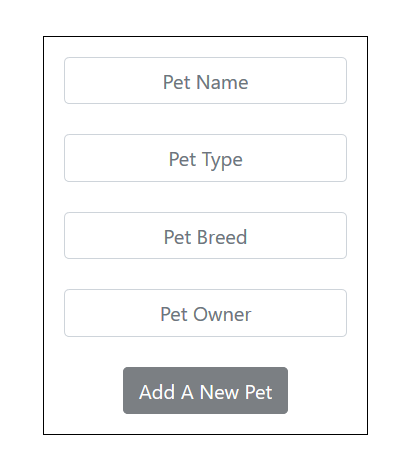
## 3.7 – Insert PHP (Pets and Guitars)

The next step is to insert new data into a PHP table using a very similar method which was just described above, with the only difference being the file that gets called with the *getJSON* method. This new file which is called by the *getJSON* is *php-insert.php.* To start off this extra function call, it needs to test the connection of the file and follow similar steps as stated above.

But as a recap it follows this method, test syntax if no errors, include the database information file (*dbinfo.info.php*) and test the database connection using the information which was supplied in the dbinfo file. Build the PDO and check for errors, if no errors, get the tableName and append the data which was supplied in the previous file. Start to build the query before getting the values from the previous file again and count how many records are being inserted, before executing.

All of the JavaScript up to now, is contained with the JavaScript file called *myjs.js*. There were some other JavaScript and jQuery functions being called within the main *index.html* file. This was due to a few bugs with them cancelled each other out if they were in a separate file.

## 3.8 – Forms

For this part, the project spec states to hide additional stuff and only show it when a button is pressed. This could have been implemented in multiple ways, but for this example a lot of HTML is built but hidden to start with. For example, within this example there is an extra form which appears whenever a table is displayed. Each of these tables have their own unique form which is used to allow the user to easily insert their own data.

Seeing as each table required a certain set of data types to be added, for this function to be robust there had to be a form for each table. This required having to create 4 forms and link them up to the insert methods jQuery call which was taking their values and syncing them up. For example;

*formData.PETNAME = $('#xmlPetName').val();*

Also take note that the *Add A New Pet* button is slightly greyed out, this is due to some jQuery which is disabling the button until all the required fields have some sort of data within them. This was implemented to avoid having blank entries being sent to the XML and PHP databases.

After setting up one form which worked and was robust, it was then time to recreate it for the 3 other tables, which is a lot of code but little work to do to link them up with different values. Due to having to create 4 different forms, if they weren’t hidden to start with, they would always be on the main page. Therefore, they were hidden and only activated when the respected button was clicked. For example, whenever someone clicks XML Pets, the pet table appears, the insert xml form appears and so does a search, which will be covered next.

## 3.8 – Search

To create a search function required a lot of testing and a lot of jQuery calls being implemented. Whilst a search for a website seems like such a basic task, implementing one which works well with an already created website was quite rewarding.

To start with the search form jQuery, there needs to be a div which allows the user to type in, this is the div with search\_field as the id. This input text field allows the user to type, which then triggers the function to start.

Searching through all the table data for value which was entered the search\_field box, leaves the user with less data to look through within the table.

# 4 - PDO

The PHP documentation describes PDO as something which represents a connection between PHP and a database server (PHP, 2020). Which is almost like a variable that stores all your database information that can be used to carry out a bunch of commands which you pass it. Within this website, there was use of PDO to set our SQL within a variable that will ready the command, execute the command and even give error information.

Within this website, the PDO was started off as a PDO::\_\_construct which is what stores our database details. Below is the PDO variable used within the project, note it doesn’t give out any of our personal information.

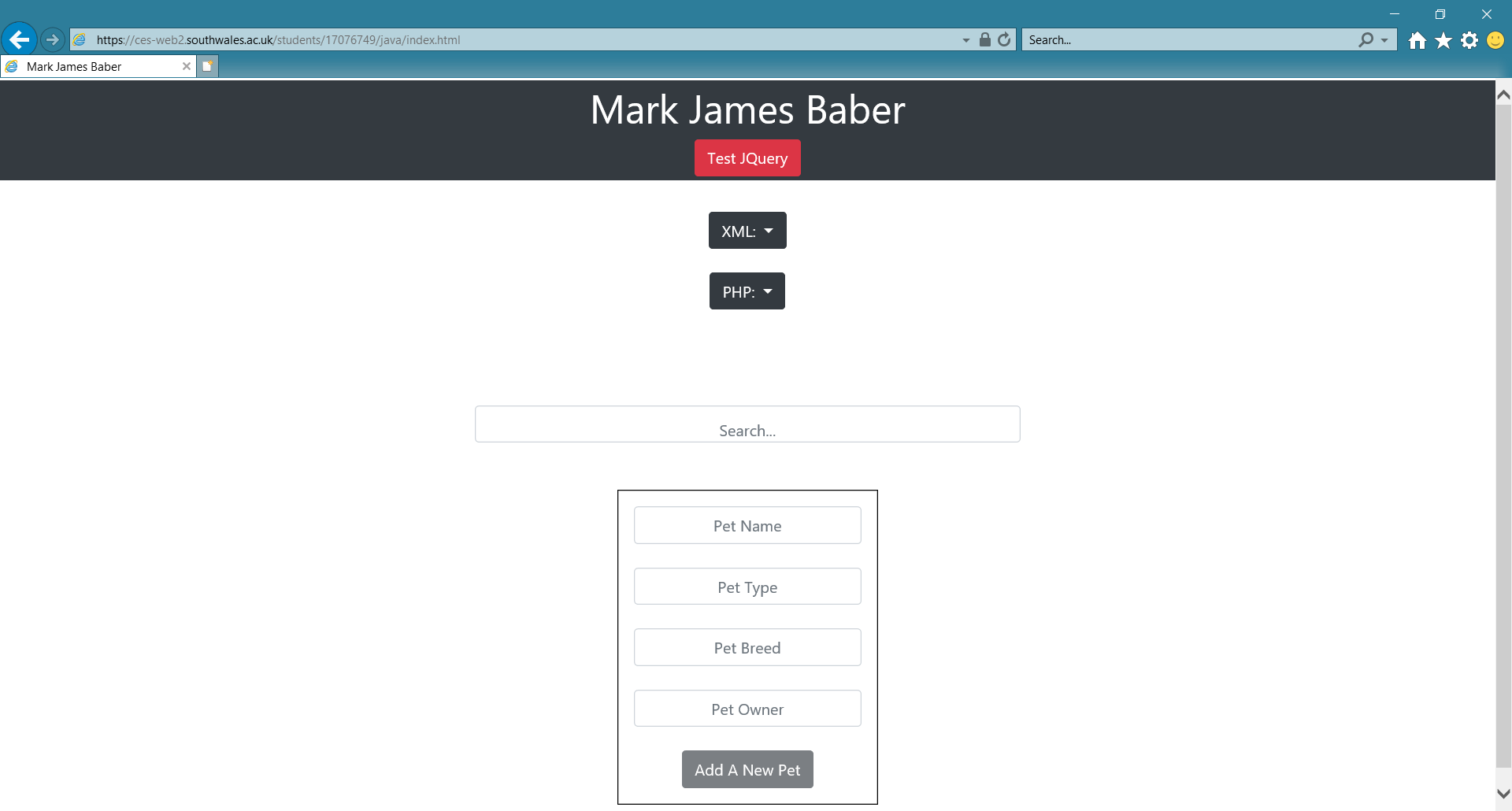
*$pdo = new PDO("mysql:host=$host;dbname=$database;charset=utf8", $username, $password, [PDO:: ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION, PDO::ATTR\_EMULATE\_PREPARES => false]);*

The next part of the PDO is the prepare section which can have SQL stored within, this is useful as it doesn’t allow people to use SQL injection within the browser. After preparing the PDO to carry out the required SQL, the user can then use the command execute to start off the process.

SQL Injection is carried out when a user can type an SQL command within the URL to get or delete your database tables without the need to login to it.

# 5 – Testing

Testing this website with different browsers is import, especially if this website was going to be used within the work place due to so many companies relying on old software like Internet Explorer. On first use with Internet Explorer the test jQuery function doesn’t work, the table doesn’t display with the use of jQuery. But the rest of the website seems to be displayed correctly which was surprising.



When testing with the browsers Firefox and Google Chrome, all the website worked as predicted which isn’t a shock from the 2 most popular browser.

# 6 – Conclusion

In conclusion after doing this website, there was a lot to learn by doing this project. The skills developed include, JavaScript, jQuery and the use of PDO. For someone to tackle this without much experience with JavaScript would prove to be quite difficult, but after a lot of research and the right kind of thinking about would be able to pick this us. This report outlines some of the more difficult parts of taking on a project which requires a lot of jQuery to add new data to 2 different file types, which are XML files which are local to the website and PHP which is external to this website.

# 7 - References

*Bootstrap* (2019) Available at: <https://getbootstrap.com/docs/4.4/getting-started/introduction/> (Accessed 05/12/2019).

*W3schools* (2019) Available at: <https://www.w3schools.com/howto/howto_js_dropdown.asp> (Accessed 06/12/2019)

*jQuery* (2019) Available at: <https://api.jquery.com/> (Accessed 06/12/2019)

*PHP* (2020) Available at: <https://www.php.net/manual/en/class.pdo.php> (Accessed 07/01/2020)

# 8 - Software

Windows 10 – 1909 (OS Build 18363.535)

Firefox - 72.0.1 (64-bit) – Windows

Google Chrome - Version 79.0.3945.117 (Official Build) (64-bit) – Windows

Visual Studio Code – 1.41.1

XAMPP – 3.2.4

GitHub Desktop – 2.2.4

# 9 – Appendix

## 9.1 – Index.html

<!DOCTYPE html>

<html lang="en">

<head>

<!-- Required meta tags -->

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<!-- JavaScript Ref -->

<script src="myjs.js"></script>

<!-- CSS Ref -->

<link rel="stylesheet" href="css/mycss.css">

<!-- Bootstrap CSS -->

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"

integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">

<!-- Bootstrap JS-->

<script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"

integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n" crossorigin="anonymous">

</script>

<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"

integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous">

</script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"

integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous">

</script>

<!-- JQuery From Google -->

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<title>Mark James Baber</title>

</head>

<body>

<script>

window.jQuery || document.write('<script src="jquery-3.3.1.min.js "><\/script>')

</script>

<div class="jumbotron jumbotron-fluid text-center p-1"

style="background-color: #343a40; border-top: solid 2px #fff;">

<div class="container">

<h1 style="color:#fff">Mark James Baber</h1>

<button class="btn btn-danger" onclick="testJQ()">Test JQuery</button>

</div>

</div>

<div class="container text-center">

<div class="dropdown">

<button class="btn btn-dark dropdown-toggle" type="button" data-toggle="dropdown"

onclick="dropDownXML();">XML:

<span class="caret"></span></button>

<ul class="dropdown-menu" id="dropdown-xml">

<li><a class="dropdown-item" id="xml-btn-pets" onclick="xmlPets();">Pets</a></li>

<div class="dropdown-divider"></div>

<li><a class="dropdown-item" id="xml-btn-guitars" onclick="xmlGuitars();">Guitars</a></li>

</ul>

</div>

</div>

<br>

<div class="container text-center">

<div class="dropdown">

<button class="btn btn-dark dropdown-toggle" type="button" data-toggle="dropdown"

onclick="dropDownPHP();">PHP:

<span class="caret"></span></button>

<ul class="dropdown-menu" id="dropdown-php">

<li><a class="dropdown-item" id="php-btn-pets" onclick="phpPets();">Pets</a></li>

<div class="dropdown-divider"></div>

<li><a class="dropdown-item" id="php-btn-guitars" onclick="phpGuitars();">Guitars</a></li>

</ul>

</div>

</div>

<br>

<br>

<div class="container text-center">

<table class="center" id="dataTable"></table>

<br>

<div class="success text-center"></div>

<br>

<input type="text" placeholder="Search..." id="search\_field" class="form-control text-center">

<br>

<form class="text-center" id="insertForm">

<br>

<!-- XML PETS Form -->

<table class="center" id="xmlPets">

<tr>

<td colspan="4" class="text-center">

<input type='text' class="form-control text-center" id='xmlPetName' placeholder='Pet Name'

autocomplete="off" required />&nbsp;

<input type='text' class="form-control text-center" id='xmlPetType' placeholder='Pet Type'

autocomplete="off" required />&nbsp;

<input type='text' class="form-control text-center" id='xmlPetBreed' placeholder='Pet Breed'

autocomplete="off" required />&nbsp;

<input type='text' class="form-control text-center" id='xmlPetOwner' placeholder='Pet Owner'

autocomplete="off" required />&nbsp;

<br>

<input type='button' class="btn btn-dark" id='xmlPetsBtn' value='Add a new pet'

disabled onclick="xmlPetsInsert()" /></td>

</tr>

</table>

<table class="center" id="xmlGuitars">

<tr>

<td colspan="4" class="text-center">

<input type='text' class="form-control text-center" id='xmlGuitarBrand'

placeholder='Guitar Brand' autocomplete="off" required />&nbsp;

<input type='text' class="form-control text-center" id='xmlGuitarModel'

placeholder='Guitar Model' autocomplete="off" required />&nbsp;

<input type='text' class="form-control text-center" id='xmlGuitarType' placeholder='Guitar Type'

autocomplete="off" required />&nbsp;

<br>

<input type='button' class="btn btn-dark" id='xmlGuitarsBtn' value='Add a new guitar'

disabled onclick="xmlGuitarsInsert()" /></td>

</tr>

</table>

<table class="center" id="phpPets">

<tr>

<td colspan="4" class="text-center">

<input type='text' class="form-control text-center" id='phpPetName' placeholder='Pet Name'

autocomplete="off" required />&nbsp;

<input type='text' class="form-control text-center" id='phpPetType' placeholder='Pet Type'

autocomplete="off" required />&nbsp;

<input type='text' class="form-control text-center" id='phpPetBreed' placeholder='Pet Breed'

autocomplete="off" required />&nbsp;

<input type='text' class="form-control text-center" id='phpPetOwner' placeholder='Pet Owner'

autocomplete="off" required />&nbsp;

<br>

<input type='button' class="btn btn-dark" id='phpPetsBtn' value='Add a new pet'

disabled onclick="phpPetsInsert()" /></td>

</tr>

</table>

<table class="center" id="phpGuitars">

<tr>

<td colspan="4" class="text-center">

<input type='text' class="form-control text-center" id='phpGuitarBrand'

placeholder='Guitar Brand' autocomplete="off" required />&nbsp;

<input type='text' class="form-control text-center" id='phpGuitarModel'

placeholder='Guitar Model' autocomplete="off" required />&nbsp;

<input type='text' class="form-control text-center" id='phpGuitarType' placeholder='Guitar Type'

autocomplete="off" required />&nbsp;

<br>

<input type='button' class="btn btn-dark" id='phpGuitarsBtn' value='Add a new guitar'

disabled onclick="phpGuitarsInsert()" /></td>

</tr>

</table>

<script>

// Start by hiding the forms

$("table#xmlPets").hide();

$("table#xmlGuitars").hide();

$("table#phpPets").hide();

$("table#phpGuitars").hide();

// Hide Search

$("#search\_field").hide();

// Function to show XML Pets Form

$("#xml-btn-pets").click(function () {

$("table#xmlPets").show();

$("table#xmlGuitars").hide();

$("table#phpPets").hide();

$("table#phpGuitars").hide();

$("#search\_field").show();

console.log('Show xml pets && hide everything else');

});

// Function to show XML Guitar Form

$("#xml-btn-guitars").click(function () {

$("table#xmlPets").hide();

$("table#xmlGuitars").show();

$("table#phpPets").hide();

$("table#phpGuitars").hide();

$("#search\_field").show();

console.log('Show xml guitars && hide everything else');

});

// Function to show php Pets Form

$("#php-btn-pets").click(function () {

$("table#xmlPets").hide();

$("table#xmlGuitars").hide();

$("table#phpPets").show();

$("table#phpGuitars").hide();

$("#search\_field").show();

console.log('Show php pets && hide everything else');

});

// Function to show php Guitar Form

$("#php-btn-guitars").click(function () {

$("table#xmlPets").hide();

$("table#xmlGuitars").hide();

$("table#phpPets").hide();

$("table#phpGuitars").show();

$("#search\_field").show();

console.log('Show php guitars && hide everything else');

});

// SEARCH FORM JQUERY

$('#search\_field').on('keyup', function () {

var value = $(this).val();

var patt = new RegExp(value, "i");

$('#dataTable').find('tr').each(function () {

if (!($(this).find('td').text().search(patt) >= 0)) {

$(this).not('td').hide();

}

if (($(this).find('td').text().search(patt) >= 0)) {

$(this).show();

}

});

});

// ENABLE BUTTONS WHEN ALL FIELDS ARE FILLED IN

// XML Pets

$('#xmlPetName, #xmlPetType, #xmlPetBreed, #xmlPetOwner').bind('keyup', function () {

if (xmlPetsFilled()) {

$('#xmlPetsBtn').removeAttr('disabled');

console.log('Button enabled');

} else {

console.log('Please fill out all fields');

}

});

// xmlPetsFilled

function xmlPetsFilled() {

var filled = true;

$('#xmlPets input').each(function () {

if ($(this).val() == '') filled = false;

});

return filled;

};

// XML Guitars

$('#xmlGuitarBrand, #xmlGuitarModel, #xmlGuitarType').bind('keyup', function () {

if (xmlGuitarsFilled()) {

$('#xmlGuitarsBtn').removeAttr('disabled');

console.log('Button enabled');

} else {

console.log('Please fill out all fields');

}

});

// xmlGuitarsFilled

function xmlGuitarsFilled() {

var filled = true;

$('#xmlGuitars input').each(function () {

if ($(this).val() == '') filled = false;

});

return filled;

};

// PHP Pets

$('#phpPetName, #phpPetType, #phpPetBreed, #phpPetOwner').bind('keyup', function () {

if (phpPetsFilled()) {

$('#phpPetsBtn').removeAttr('disabled');

console.log('Button enabled');

} else {

console.log('Please fill out all fields');

}

});

// phpPetsFilled

function phpPetsFilled() {

var filled = true;

$('#phpPets input').each(function () {

if ($(this).val() == '') filled = false;

});

return filled;

};

// PHP Guitars

$('#phpGuitarBrand, #phpGuitarModel, #phpGuitarType').bind('keyup', function () {

if (phpGuitarsFilled()) {

$('#phpGuitarsBtn').removeAttr('disabled');

console.log('Button enabled');

} else {

console.log('Please fill out all fields');

}

});

// phpGuitarsFilled

function phpGuitarsFilled() {

var filled = true;

$('#phpGuitars input').each(function () {

if ($(this).val() == '') filled = false;

});

return filled;

};

</script>

</form>

<br>

</div>

</body>

</html>

## 9.2 – CSS

\* {

text-transform: capitalize;

}

/\* Dropdown Button \*/

.dropbtn {

background-color: #3498DB;

color: white;

padding: 16px;

font-size: 16px;

border: none;

cursor: pointer;

}

/\* Dropdown button on hover & focus \*/

.dropbtn:hover,

.dropbtn:focus {

background-color: #2980B9;

}

/\* The container <div> - needed to position the dropdown content \*/

.dropdown {

position: relative;

display: inline-block;

}

/\* Dropdown Content (Hidden by Default) \*/

.dropdown-content {

display: none;

position: absolute;

background-color: #f1f1f1;

min-width: 160px;

box-shadow: 0px 8px 16px 0px rgba(0, 0, 0, 0.2);

z-index: 1;

}

/\* Links inside the dropdown \*/

.dropdown-content a {

color: black;

padding: 12px 16px;

text-decoration: none;

display: block;

}

/\* Change color of dropdown links on hover \*/

.dropdown-content a:hover {

background-color: #ddd

}

/\* Show the dropdown menu (use JS to add this class to the .dropdown-content container when the user clicks on the dropdown button) \*/

.show {

display: block;

}

/\* Center the table \*/

table.center {

margin-left:auto;

margin-right:auto;

}

table,

th,

td {

border: 1px solid black;

border-collapse: collapse;

}

th,

td {

padding: 1em;

}

#search\_field{

max-width: 50%;

padding: 1em;

margin-left:auto;

margin-right:auto;

}

.success{

max-width: 50%;

padding: 1em;

margin-left:auto;

margin-right:auto;

}

.success {

color: #4F8A10;

background-color: #DFF2BF;

display: none;

}

## 9.3 – JavaScript

// Test if JQuery is enabled

function testJQ() {

if (window.jQuery) {

// jQuery is loaded

alert("jQuery loaded correctly!");

} else {

// jQuery is not loaded

alert("JQuery isn't loaded correctly!");

}

}

// ###############################################################################################################################################

// ###############################################################################################################################################

// Dropdown JQuery Effects

function dropDownXML() {

$('#dropdown-xml').hide().delay(100).slideDown(250);

};

function dropDownPHP() {

$('#dropdown-php').hide().delay(100).slideDown(250);

}

// ###############################################################################################################################################

// ###############################################################################################################################################

// GET PETS XML

function xmlPets() {

$(function () {

$.getJSON("xml-get.php", {

sourceName: "pets"

}, function (data) {

if (data["code"] == "error") {

console.log(data["message"]);

} else {

$.each(data, function (index, element) {

// Set variables for Loop

var table = '<thead><tr><th scope="col">Pet Name</th><th scope="col">Pet Type</th><th scope="col">Pet Breed</th><th scope="col">Pet Owner</th></tr></thead><tbody>';

// set forEach

Object.entries(data).forEach(entry => {

let key = entry[0]; // Gets Animal

let value = entry[1] // Gets array of petNo, petName etc

});

// Traditional Loop Start

let entries = Object.entries(data);

for (let i = 0; i < entries.length; i++) {

let key = entries[i][0]; // Gets Animal

let value = entries[i][1]; // Gets array of petNo, petName etc

// Nested Loop Start

for (let j = 0; j < value.length; j++) {

table += '<tr><td>' +

(value[j].PETNAME) + "</td><td>" +

(value[j].PETTYPE) + "</td><td>" +

(value[j].PETBREED) + "</td><td>" +

(value[j].PETOWNER);

} // Nested Loop Stop

}; // Traditional Loop Stop

$('#dataTable').html(table);

});

} //end else

}); //end getJSON

});

};

// GET GUITARS XML

function xmlGuitars() {

$(function () {

$.getJSON("xml-get.php", {

sourceName: "guitars"

}, function (data) {

if (data["code"] == "error") {

console.log(data["message"]);

} else {

$.each(data, function (index, element) {

// Set variables for Loop

var i;

var table = '<thead><tr><th scope="col">Guitar Brand</th><th scope="col">Guitar Model</th><th scope="col">Guitar Type</th></thead><tbody>';

// set forEach

Object.entries(data).forEach(entry => {

let key = entry[0]; // Gets INSTRUMENTS

let value = entry[1] // Gets array of guitarNo etc

});

// Traditional Loop Start

let entries = Object.entries(data);

for (let i = 0; i < entries.length; i++) {

let key = entries[i][0]; // Gets INSTRUMENTS

let value = entries[i][1]; // Gets array of guitarNo etc

// Nested Loop Start

for (let j = 0; j < value.length; j++) {

table += '<tr><td>' +

(value[j].GUITARBRAND) + "</td><td>" +

(value[j].GUITARMODEL) + "</td><td>" +

(value[j].GUITARTYPE);

} // Nested Loop Stop

}; // Traditional Loop Stop

$('#dataTable').html(table);

});

} //end else

}); //end getJSON

});

};

// ###############################################################################################################################################

// ###############################################################################################################################################

// GET Pets PHP

function phpPets() {

$(function () {

$.getJSON("php-get.php", {

tableName: "pets"

}, function (data) {

if (data["code"] == "error") {

console.log(data["message"]);

} else {

$.each(data, function (index, element) {

// Set variables for Loop

var i;

var table = '<thead><tr><th scope="col">Pet Name</th><th scope="col">Pet Type</th><th scope="col">Pet Breed</th><th scope="col">Pet Owner</th></tr></thead><tbody>';

// set forEach

Object.entries(data).forEach(entry => {

let key = entry[0]; // Gets INSTRUMENTS

let value = entry[1] // Gets array of guitarNo etc

});

// Traditional Loop Start

let entries = Object.entries(data);

for (let i = 0; i < entries.length; i++) {

let key = entries[i][0]; // Gets INSTRUMENTS

let value = entries[i][1]; // Gets array of guitarNo etc

// Nested Loop Start

for (let j = 0; j < value.length; j++) {

table += '<tr><td>' +

(value[j].petName) + "</td><td>" +

(value[j].petType) + "</td><td>" +

(value[j].petBreed) + "</td><td>" +

(value[j].petOwner);

} // Nested Loop Stop

}; // Traditional Loop Stop

$('#dataTable').html(table);

});

}

})

})

};

// GET Guitars PHP

function phpGuitars() {

$(function () {

$.getJSON("php-get.php", {

tableName: "guitars"

}, function (data) {

if (data["code"] == "error") {

console.log(data["message"]);

} else {

$.each(data, function (index, element) {

// Set variables for Loop

var i;

var table = '<thead><tr><th scope="col">Guitar Brand</th><th scope="col">Guitar Model</th><th scope="col">Guitar Type</th></thead><tbody>';

// set forEach

Object.entries(data).forEach(entry => {

let key = entry[0]; // Gets INSTRUMENTS

let value = entry[1] // Gets array of guitarNo etc

});

// Traditional Loop Start

let entries = Object.entries(data);

for (let i = 0; i < entries.length; i++) {

let key = entries[i][0]; // Gets INSTRUMENTS

let value = entries[i][1]; // Gets array of guitarNo etc

// Nested Loop Start

for (let j = 0; j < value.length; j++) {

table += '<tr><td>' +

(value[j].guitarBrand) + "</td><td>" +

(value[j].guitarModel) + "</td><td>" +

(value[j].guitarType);

} // Nested Loop Stop

}; // Traditional Loop Stop

$('#dataTable').html(table);

});

}

})

})

};

// INSERT PETS XML

function xmlPetsInsert() {

// Create the data which will be added into the database

var formData = new Object();

formData.PETNAME = $('#xmlPetName').val();

formData.PETTYPE = $('#xmlPetType').val();

formData.PETBREED = $('#xmlPetBreed').val();

formData.PETOWNER = $('#xmlPetOwner').val();

// Add to JSON

console.log(formData);

var jsonFormData = JSON.stringify(formData);

console.log(jsonFormData);

var sourceFile = 'pets';

// Get the json we want to save to

$.getJSON("xml-insert.php", {

sourceName: sourceFile,

sourceData: jsonFormData,

success: function () {

$('.success').show(2000).html("Record inserted correctly").delay(1000).fadeOut(1000);

}

},

function (data) {

console.log(data);

});

};

// INSERT GUITARS XML

function xmlGuitarsInsert() {

// Create the data which will be added into the database

var formData = new Object();

formData.GUITARBRAND = $('#xmlGuitarBrand').val();

formData.GUITARMODEL = $('#xmlGuitarModel').val();

formData.GUITARTYPE = $('#xmlGuitarType').val();

// Add to JSON

console.log(formData);

var jsonFormData = JSON.stringify(formData);

console.log(jsonFormData);

var sourceFile = 'guitars';

// Get the json we want to save to

$.getJSON("xml-insert.php", {

sourceName: sourceFile,

sourceData: jsonFormData,

success: function () {

$('.success').show(2000).html("Record inserted correctly").delay(1000).fadeOut(1000);

}

},

function (data) {

console.log(data);

});

};

// INSERT PETS PHP

function phpPetsInsert() {

// Create the data which will be added into the database

var addData = new Object();

addData.petName = $('#phpPetName').val();

addData.petType = $('#phpPetType').val();

addData.petBreed = $('#phpPetBreed').val();

addData.petOwner = $('#phpPetOwner').val();

// Add to JSON

var jsonAddData = JSON.stringify(addData);

// Get the json we want to save to

$.getJSON("php-insert.php", {

tableName: "pets",

appendData: jsonAddData,

success: function () {

$('.success').show(2000).html("Record inserted correctly").delay(1000).fadeOut(1000);

}

},

function (data) {

console.log(data);

});

};

// INSERT GUITARS PHP

function phpGuitarsInsert() {

// Create the data which will be added into the database

var addData = new Object();

addData.guitarBrand = $('#phpGuitarBrand').val();

addData.guitarModel = $('#phpGuitarModel').val();

addData.guitarType = $('#phpGguitarType').val();

// Add to JSON

var jsonAddData = JSON.stringify(addData);

// Get the json we want to save to

$.getJSON("php-insert.php", {

tableName: "guitars",

appendData: jsonAddData,

success: function () {

$('.success').show(2000).html("Record inserted correctly").delay(1000).fadeOut(1000);

}

},

function (data) {

console.log(data);

});

};

## 9.4 – php-get.php

<?php

if (!isset($error)) {

$error = new stdClass();

}

// Get details from dbinfo

include "dbinfo.info.php";

try {

$pdo = new PDO("mysql:host=$host;dbname=$database;charset=utf8", $username, $password, [PDO:: ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION, PDO::ATTR\_EMULATE\_PREPARES => false]);

} catch (PDOException $e) {

$error->code = "error";

$error->message = "There was a problem connecting to the database";

echo json\_encode($error);

$pdo = null;

return;

}

$table = $\_GET["tableName"];

$stmt = $pdo->prepare("SELECT \* FROM {$table}");

$result = $stmt->execute();

if ($stmt->rowCount() > 0)

{

$tableData = array();

$tableData[] = $stmt->fetchAll(PDO::FETCH\_ASSOC);

echo json\_encode($tableData);

}

else

{

$error->code = "error";

$error->message = "The table: ".$table." contains no rows.";

echo json\_encode($error);

}

$stmt = null;

$pdo = null;

?>

## 9.5 – php-insert.php

<?php

if (!isset($error)) {

$error = new stdClass();

}

include "dbinfo.info.php";

try {

///$dbh = new PDO("mysql:host=$host;dbname=$database;charset=utf8", $username, $password);

$pdo = new PDO("mysql:host=$host;dbname=$database;charset=utf8", $username, $password, [PDO:: ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION, PDO::ATTR\_EMULATE\_PREPARES => false]);

// check database connection

} catch (PDOException $e) {

$error->code = "error";

$error->message = "There was a problem connecting to the database";

echo json\_encode($error);

$pdo = null;

return;

}

$table = $\_GET["tableName"];

$appendData = $\_GET['appendData'];

$json\_array = json\_decode($appendData, true);

$query = "INSERT INTO ".$table." (";

$placeholder = "";

$values = array();

foreach($json\_array as $key => $value) {

$query .= $key.", ";

$placeholder .= "?, ";

$values[] = $value;

}

$query = rtrim($query, ', ');

$placeholder = rtrim($placeholder, ', ');

$query .= ") VALUES (".$placeholder.")";

$stmt = $pdo->prepare($query);

$counter = 1;

foreach ($json\_array as $key => &$val) {

$stmt->bindParam($counter, $val);

$counter++;

}

$stmt->execute();

echo $pdo->lastInsertId();

$stmt = null;

$pdo = null;

?>

## 9.6 – xml-get.php

<?php

libxml\_use\_internal\_errors(true);

libxml\_clear\_errors();

if (!isset($error)){

$error = new stdClass();

}

# Get The SourceName from the html file

if(!isset($\_GET["sourceName"])){

$error->code ="error";

$error->message = "no file name";

echo json\_encode($error);

return;

}

# The file name is called sourceName in html

$filename = $\_GET["sourceName"];

if (!file\_exists("xml/".$filename.".xml")){

$error->code ="error";

$error->message = "no such file";

echo json\_encode($error);

return;

}

$xmlFile = simplexml\_load\_file("xml/".$filename.".xml");

$errors = libxml\_get\_errors();

if (empty($xmlFile)){

$error->code ="error";

$error->message = "no contents";

echo json\_encode($error);

return;

}

if ($xmlFile->count() == 0){

$error->code ="error";

$error->message = "no elements";

echo json\_encode($error);

return;

}

if ($errors){

$error->code ="error";

$error->message = "no idea";

echo json\_encode($error);

return;

}

echo json\_encode($xmlFile);

?>

## 9.7 – xml-insert.php

<?php

$sData = $\_GET["sourceData"];

$filename = $\_GET["sourceName"];

$fullPath = "xml/".$filename.".xml";

$xml = simplexml\_load\_file($fullPath);

$json\_array = json\_decode($sData, true);

$element = $xml->addChild($xml->children()->getName());

foreach($json\_array as $key => $value){

$element->addChild($key, $value);

}

$dom = new DOMDocument('1.0');

$dom->preserveWhiteSpace = false;

$dom->formatOutput = true;

$dom->loadXML($xml->asXML());

$xml = new SimpleXMLElement($dom->saveXML());

$xml->saveXML("xml/".$filename.".xml");

echo json\_encode($xml);

return;

?>

## 9.8 – XML

### 9.8.1 – Guitars

<?xml version="1.0" encoding="UTF-8"?>

<INSTRUMENTS>

<GUITAR>

<GUITARBRAND>Fender</GUITARBRAND>

<GUITARMODEL>Starcaster</GUITARMODEL>

<GUITARTYPE>Electric Guitar</GUITARTYPE>

</GUITAR>

<GUITAR>

<GUITARBRAND>Squier</GUITARBRAND>

<GUITARMODEL>Jazz</GUITARMODEL>

<GUITARTYPE>Electric Bass</GUITARTYPE>

</GUITAR>

<GUITAR>

<GUITARBRAND>Gibson</GUITARBRAND>

<GUITARMODEL>Marauder</GUITARMODEL>

<GUITARTYPE>Electric Guitar</GUITARTYPE>

</GUITAR>

</INSTRUMENTS>

### 9.8.2 – Pets

<?xml version="1.0" encoding="UTF-8"?>

<PET>

<ANIMAL>

<PETNAME>Ollie</PETNAME>

<PETTYPE>Dog</PETTYPE>

<PETBREED>Border Collie</PETBREED>

<PETOWNER>Greggie G</PETOWNER>

</ANIMAL>

<ANIMAL>

<PETNAME>Max</PETNAME>

<PETTYPE>Dog</PETTYPE>

<PETBREED>Staff x Shitzu</PETBREED>

<PETOWNER>Lauren T</PETOWNER>

</ANIMAL>

<ANIMAL>

<PETNAME>Lily</PETNAME>

<PETTYPE>Cat</PETTYPE>

<PETBREED>Maine Coon</PETBREED>

<PETOWNER>Mark B</PETOWNER>

</ANIMAL>

<ANIMAL>

<PETNAME>Daisy</PETNAME>

<PETTYPE>Cat</PETTYPE>

<PETBREED>Scottish Fold</PETBREED>

<PETOWNER>Mark B</PETOWNER>

</ANIMAL>

<ANIMAL>

<PETNAME>Susan Strong</PETNAME>

<PETTYPE>Dog</PETTYPE>

<PETBREED>Samoyed</PETBREED>

<PETOWNER>Dr Thomas</PETOWNER>

</ANIMAL>

</PET>

## 9.9 – Database Backup

-- phpMyAdmin SQL Dump

-- version 4.8.5

-- https://www.phpmyadmin.net/

--

-- Host: localhost

-- Generation Time: Jan 08, 2020 at 05:42 PM

-- Server version: 8.0.15

-- PHP Version: 7.3.1

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

SET AUTOCOMMIT = 0;

START TRANSACTION;

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

--

-- Database: `db\_17076749`

--

-- --------------------------------------------------------

--

-- Table structure for table `guitars`

--

CREATE TABLE `guitars` (

`guitarNo` int(11) NOT NULL,

`guitarBrand` varchar(50) NOT NULL,

`guitarModel` varchar(50) DEFAULT NULL,

`guitarType` varchar(50) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

--

-- Dumping data for table `guitars`

--

INSERT INTO `guitars` (`guitarNo`, `guitarBrand`, `guitarModel`, `guitarType`) VALUES

(1, 'Fender', 'Starcaster', 'Electric Guitar'),

(2, 'Squier', 'Jazz', 'Electric Bass'),

(3, 'Gibson', 'Marauder', 'Electric Bass'),

(4, 'Chapman Guitars', 'ML3 Pro', 'Electric Guitar'),

(5, 'Epiphone', 'ES-339', 'Electric Guitar'),

(6, 'Gretsch', 'White Falcon', 'Electric Guitar'),

(7, 'Yamaha', 'ERG 121', 'Electric Guitar'),

(8, 'Yamaha', 'ERG 121', 'Electric Guitar');

-- --------------------------------------------------------

--

-- Table structure for table `pets`

--

CREATE TABLE `pets` (

`petNo` int(11) NOT NULL,

`petName` varchar(50) NOT NULL,

`petType` varchar(50) DEFAULT NULL,

`petBreed` varchar(50) DEFAULT NULL,

`petOwner` varchar(50) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

--

-- Dumping data for table `pets`

--

INSERT INTO `pets` (`petNo`, `petName`, `petType`, `petBreed`, `petOwner`) VALUES

(1, 'Ollie', 'Dog', 'Border Collie', 'Greg Thomas'),

(2, 'Max', 'Dog', 'Staff Shitzu', 'Lauren Thomas'),

(3, 'Lily', 'Cat', 'Maine Coon', 'Mark Baber'),

(4, 'Daisy', 'Cat', 'Munchkin', 'Mark Baber'),

(5, 'Sparks', 'Cat', 'Scottish Fold', 'Elliot Ash'),

(6, 'Pooch', 'Dog', 'Jack Russell Terrier', 'Yvonne Evans'),

(7, 'Roger', 'Rabbit', 'British Giant', 'Vivian Smith');

--

-- Indexes for dumped tables

--

--

-- Indexes for table `guitars`

--

ALTER TABLE `guitars`

ADD PRIMARY KEY (`guitarNo`);

--

-- Indexes for table `pets`

--

ALTER TABLE `pets`

ADD PRIMARY KEY (`petNo`);

--

-- AUTO\_INCREMENT for dumped tables

--

--

-- AUTO\_INCREMENT for table `guitars`

--

ALTER TABLE `guitars`

MODIFY `guitarNo` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=9;

--

-- AUTO\_INCREMENT for table `pets`

--

ALTER TABLE `pets`

MODIFY `petNo` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=9;

COMMIT;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;