Ryan Lewis

14568811

Advanced internet and mobile computing is3s664

jQuery & Data Sources

**Contents**

[Design 2](#_Toc500855369)

[JavaScript Functionality 3](#_Toc500855370)

[PDO Processing 12](#_Toc500855371)

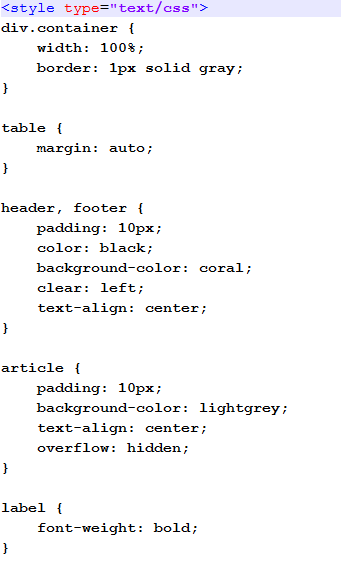
[XML Processing 18](#_Toc500855372)

[Testing different browsers 24](#_Toc500855373)

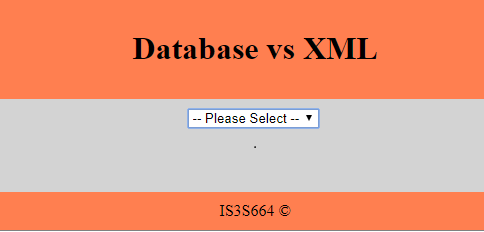
[Code Listings 28](#_Toc500855374)

# **Design**

The aim for the design of the system was to make it look simple and aesthetically pleasing while also being very easy to use.

With the use of some simple CSS (to the left) we have been able to set the layout of the webpage to be very clear and easy to read and the colour scheme is set to a dull orange and grey so that it is not too distracting to the user (seen below).

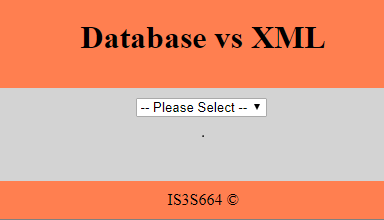
From this screenshot below and from the rest of the report you will see that the system is very self-explanatory as there is only ever one or two options for the user, making it very straightforward and easy to use for anybody who uses this system.



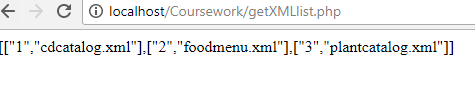
# **JavaScript Functionality**

JavaScript has mainly been used in this project to manipulate html elements such as select boxes, forms and tables. It is very useful as it can update that element dynamically without having to refresh the whole page itself. Below are some screen shots of how JavaScript has been used throughout this project.

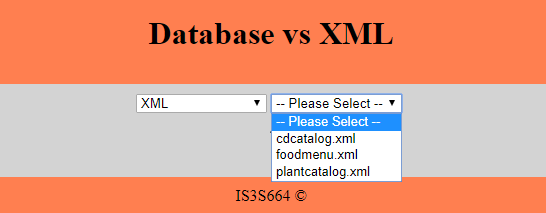
Examples

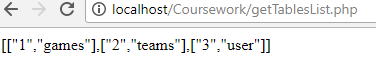
As you can see when the index.html is first loaded all that is displayed is the one select box and everything else is hidden through JavaScript’s .hide.

Then when either XML or Database is selected from the first box a php script is executed and returns a list of XML filenames in a JSON format as seen below.

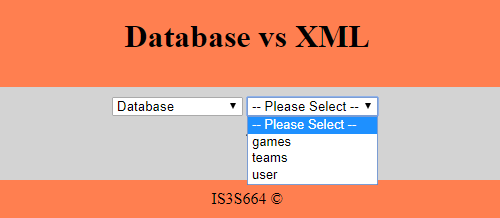


On the main page it then runs a plugin called jCombo which takes the JSON and inserts it into a html select box with the options being the filenames. And then through JavaScript’s .show it makes the box visible so the user can interact with it.

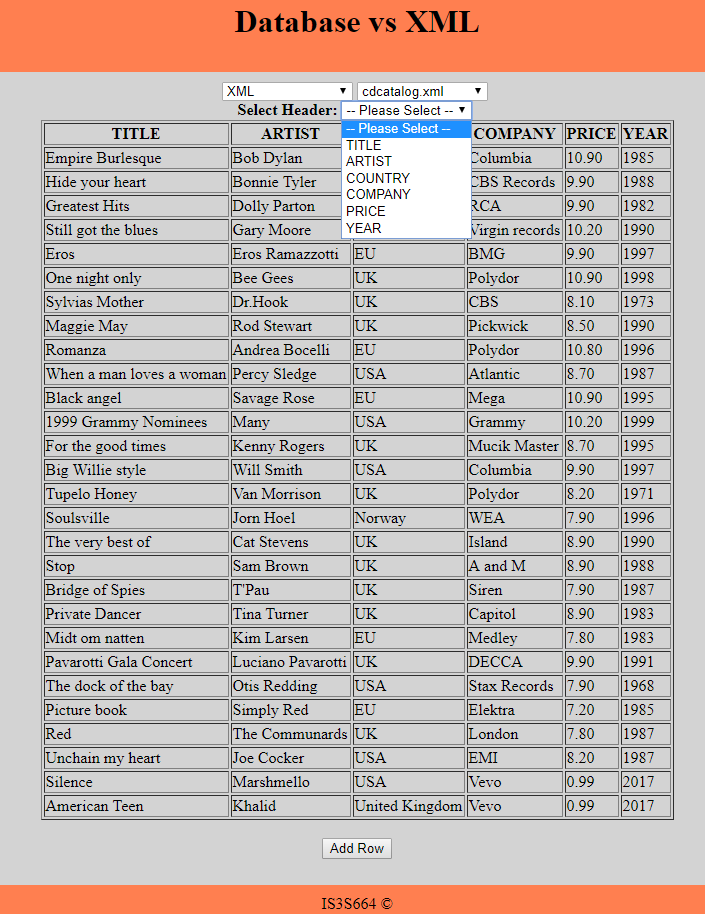




This is the example of the database side which uses the same functionality but in the php script it gets the table names from a phpMyAdmin Database and again returns them to the html in a JSON format.



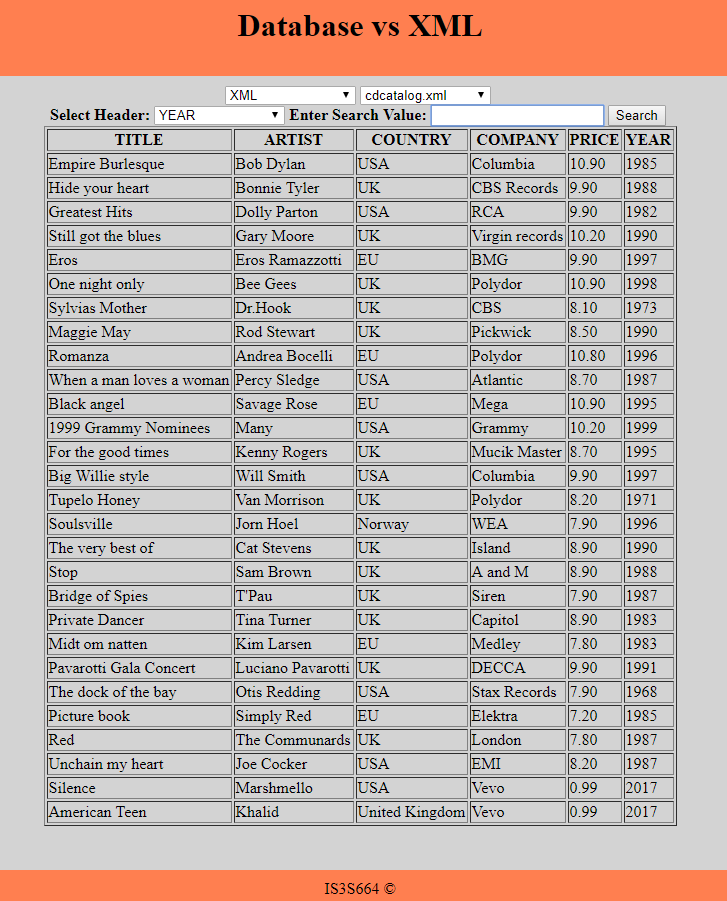


Above is what the php script returns when a filename is selected out of the XML list of files. So when a file is selected for example cdcatalog.xml it is sent to the php script via “filename” as you can see above in the url, this then get’s used to access that XML file, read it and send it back in JSON for the index to read.

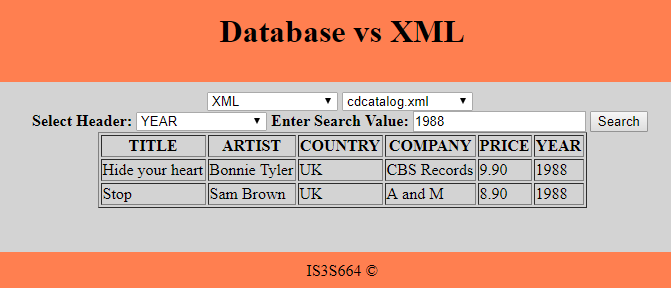
Within the index.html we use jQuery’s $getJSON feature to read the JSON and put it into a variable which then gets used in the JavaScript function called buildHTMLTable that you can see working to the left.

You can also see that there is a third drop down box that is displayed with all of the column names which is obtained in the same way as the last but the data is loaded from an XML schema instead.

When the column is selected from the options above it uses the JavaScript .show again to display a text box and also .focus to put the focus on the search box ready for the user to start typing as you can see below.



This is the result of a user typing something in and pressing the search button. This is then displayed by clearing the previous table by using .empty and displaying a new table with the search results. In the php script that searches the XML file I was only able to do it so that the search value entered must be exact to what is in the file, so for example you couldn’t type in 9 for the YEAR column and have everything with 9 in it be displayed, it is also case sensitive.

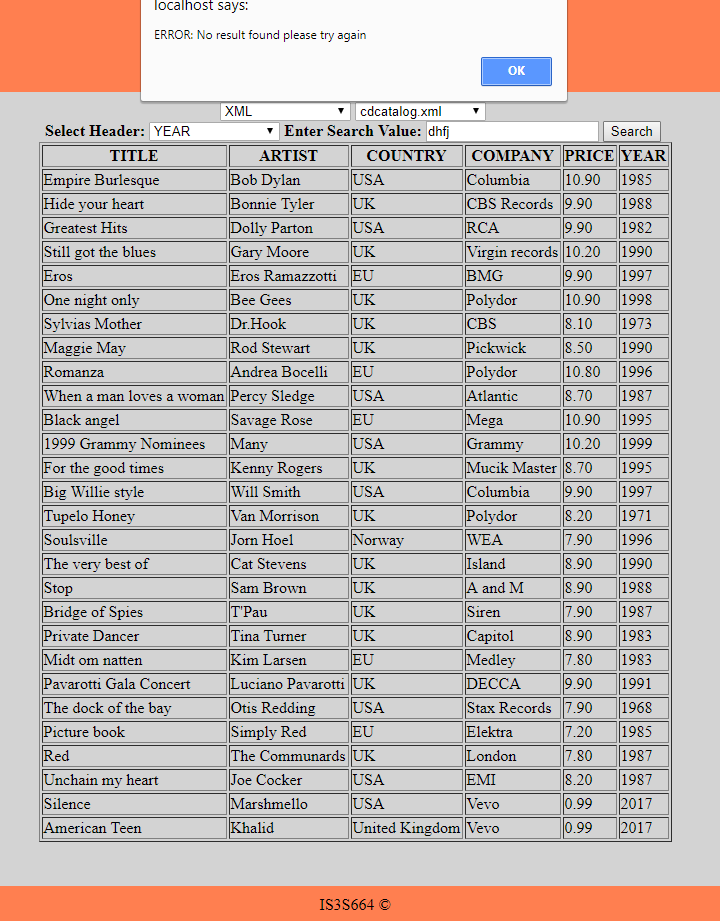


You can also see at the bottom of this page the add button is hidden to stop the user adding data while searching at the same time.

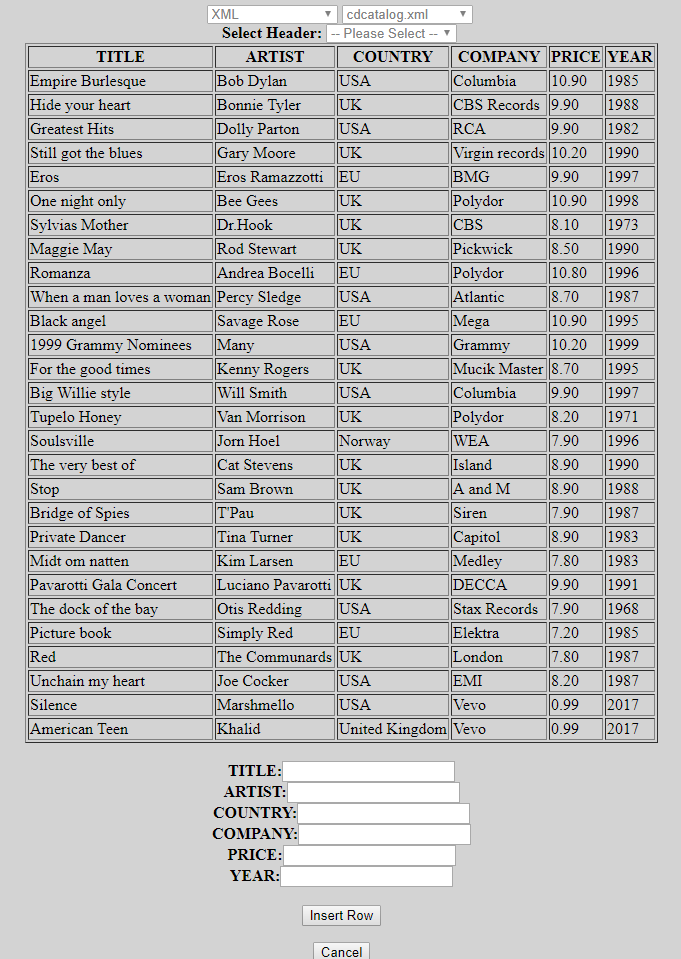
This is an example of what would happen if nothing was entered when the search button was pressed. It displays a message to let the user know that a value must be entered for the search to work and it then doesn’t complete a search and leaves the user with the full table displayed.

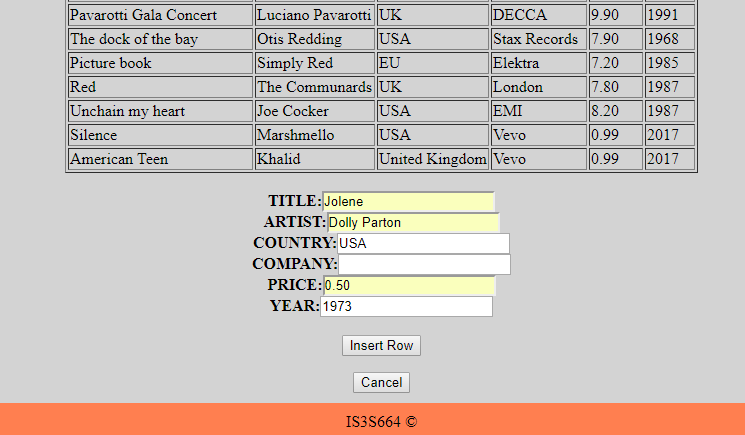


This is an example of what would happen if a value was entered but nothing was found in that column when the search button is pressed. It displays a message to let the user know that what they entered has returned 0 results and to try again, it also leaves the user with the full table displayed.



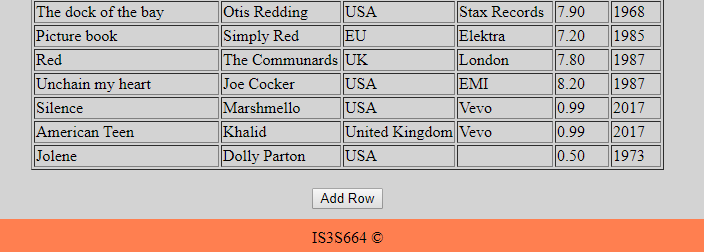
Below is what happens when the add button is pressed, first of all you can see at the top of the page the select boxes are disabled to make sure the user doesn’t do anything else while data is being added. It then displays a form ready for the user to start inputting data, this form is created by getting the column headers from the XML schema and returning it as JSON data again. This JSON is then put into a variable and used in the function called buildForm which creates this form depending on the column headers returned.





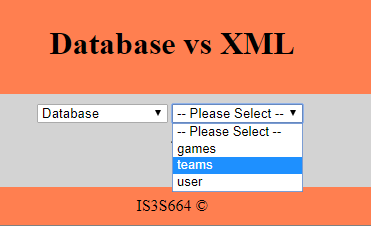
Above is data entered in the form by the user and as you can see I haven’t set any constraints on what the user can enter. The main reason being that as the form is dynamically created the input boxes are different every time and I was not able to find a way of checking what the user had entered when pressing the submit button. If I had given myself more time I would’ve liked to set it up so that there were constraints for certain fields for example a date or fields like the name/title could not be left blank. Anyway below is what happens when the insert row button is pressed, it hides the form once again and inserts the data into the XML file, the table updates automatically but only after a few seconds to allow for the XML file to update properly. This has been done with the use of setTimeout function which runs the updateTable function only after the timer has ended which is why there is a slight delay.

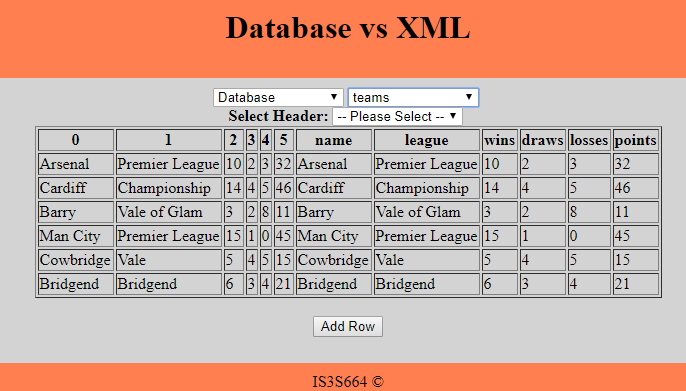
You can also see above that there is a Cancel button which if pressed will clear any data entered also hide the form and bring back the Add Row button like below. Again this is done through the use of JavaScript’s .show and .hide.

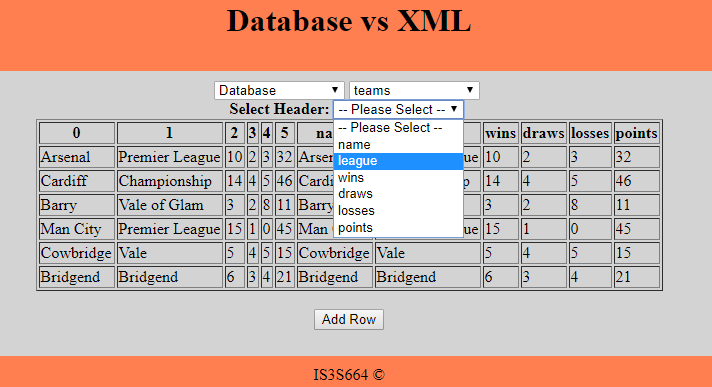


# **PDO Processing**

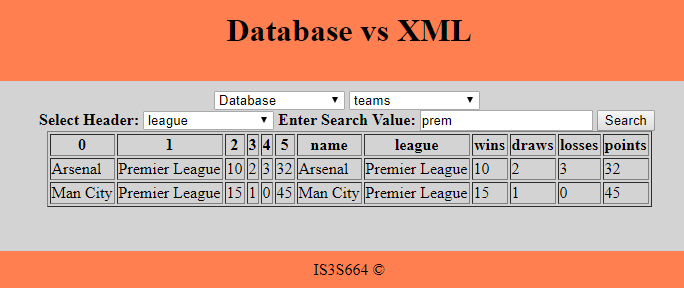
Throughout this system whenever there is a connection to the database it is done with the use of a PDO connection along with prepared SQL statements to attempt to try and prevent any SQL injection. SQL injection is possible whenever there is a user input so as you can see below with the selection menu the majority of interaction with the database is done when the parameters are set by the user’s selection and not the users input.



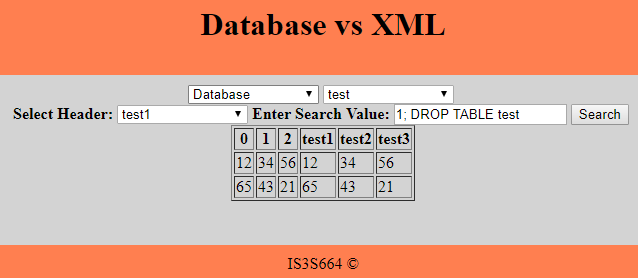




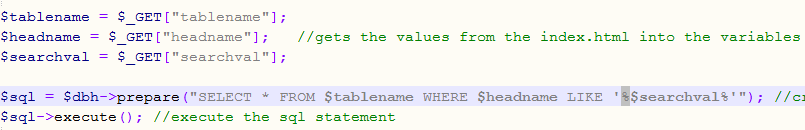
However as you can see here when the user wants to do a search they must input a value, this creates an opportunity for SQL injection. Below is how it’s supposed to work as you expect the user to enter a standard string such as this.



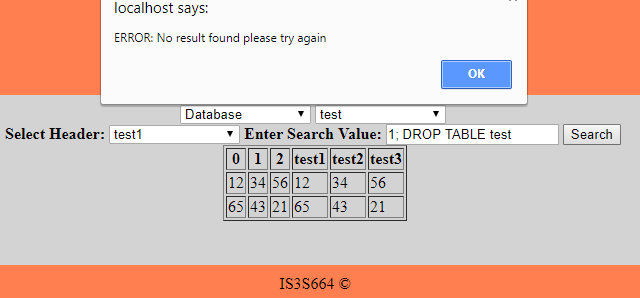
There may be some users who have a knowledge of SQL and try to enter something like this below. This string could possibly still search for 1 but then start a new query attempting to drop the table called test.



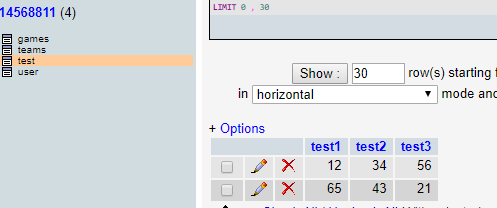
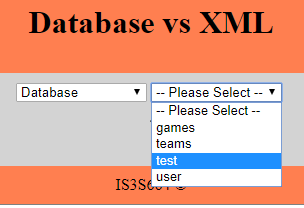
To prevent this SQL injection I have used a prepared SQL statement as you can see below. Prepared statements only replace the parameters with the values at runtime and also the values are treated literally and not as another SQL statement to be executed which helps a lot in preventing SQL injection.



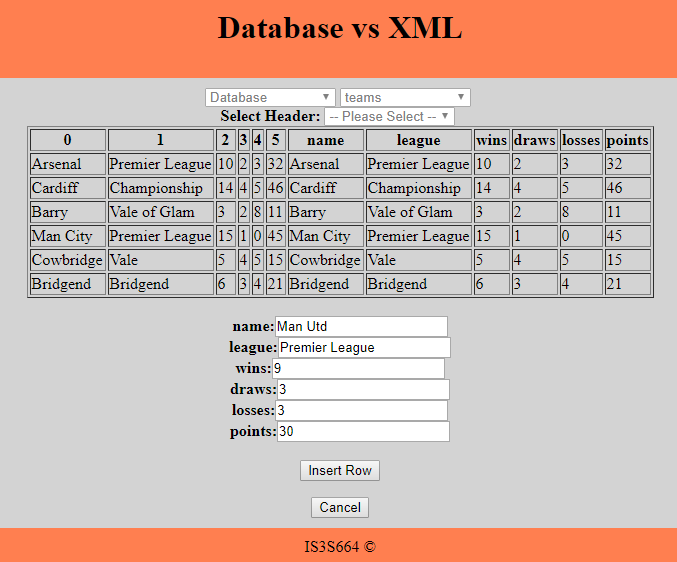
The result of this user input and SQL statement are shown below.

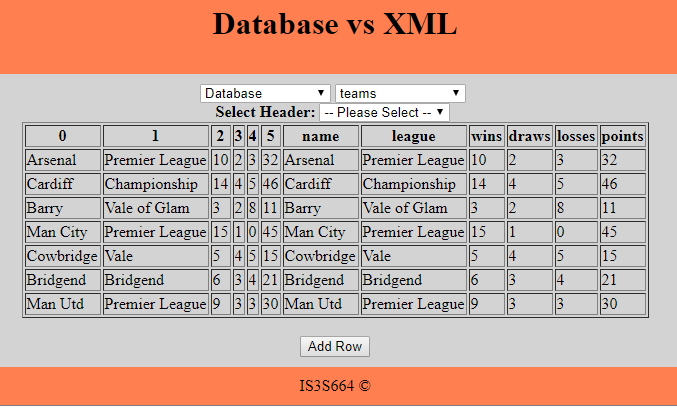


As you can see the message says that no results have been found for that search and the test table is still intact as can be seen below.

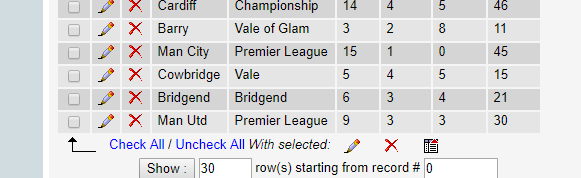


The only other input on this system is for adding a new record, below you can see it working as intended.





When the data is submitted it is entered into the database and updated on the webpage.

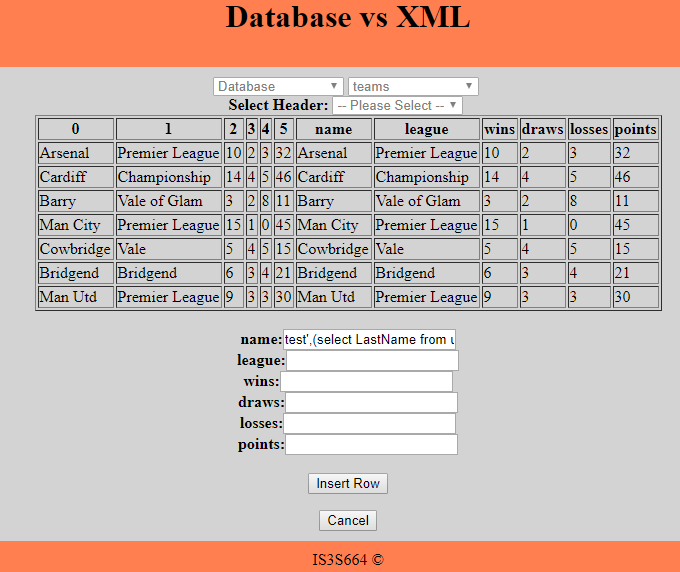


However as you can see below it only currently works for one table at a time. This is because the data sent from the same html form could be a different number of values and differently named columns depending on the table selected. Meaning that the php had to receive the table name, all the column names and the data for inserting into those columns and use them an SQL insert statement. So, as you can see below I was only able to put the data entered into the statement and the rest of the statement had to be hardcoded as I was not able to figure out a way of utilising the rest of the information in the SQL.

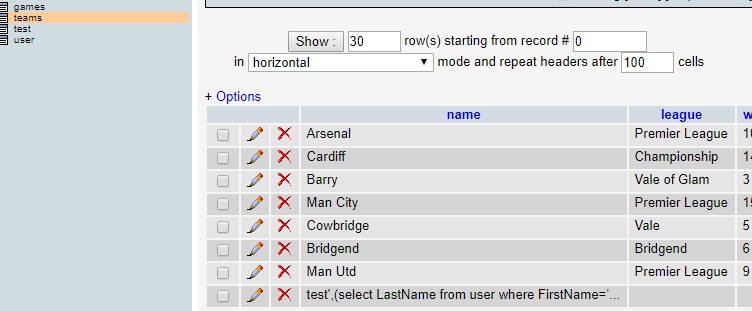


I can imagine that an insert statement is one of the most vulnerable to SQL injection and is the reason why it was very hard to use the parameters in the SQL statement. If given more time or further development in the future I would have done more research into whitelisting the parameters so the system knew they were safe, as this seemed to be the problem.

Overall this has proved that using PDO and prepared statements adds some security against SQL injection. Shown below is some more evidence of testing SQL injection on the insert. I have entered **test',(select LastName from user where FirstName=’root’ limit 0,1),’test2’)** into the text box which is attempting to get the value of last name from the table user where their first name is root.

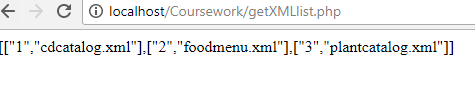


This would obviously be a problem if it was username and password in the user table and this code was to return that sensitive data. However as stated previously the prepared statement treats the parameter literally and has inserted the data that was entered straight into the name field as you can see below. It would obviously be safer to check what the user was inputting before sending it to the database to prevent certain characters being entered, but as previously mentioned I was not able to find a possible solution in the given timeframe.

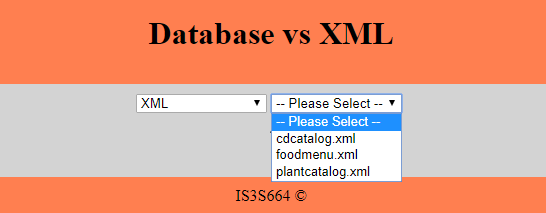


# **XML Processing**

As well as displaying database tables using PDO processing, this system has the ability to display XML tables/files. The processing for this is all done through php and the use of .xml and .xsd files, the XML files obviously contain the data and the XSD files are the XML schemas which hold information about the XML file and how it’s formatted. In this case they only contain information about the child names, however they can be used for a lot more such as limits and constraints to what can be stored in each field.



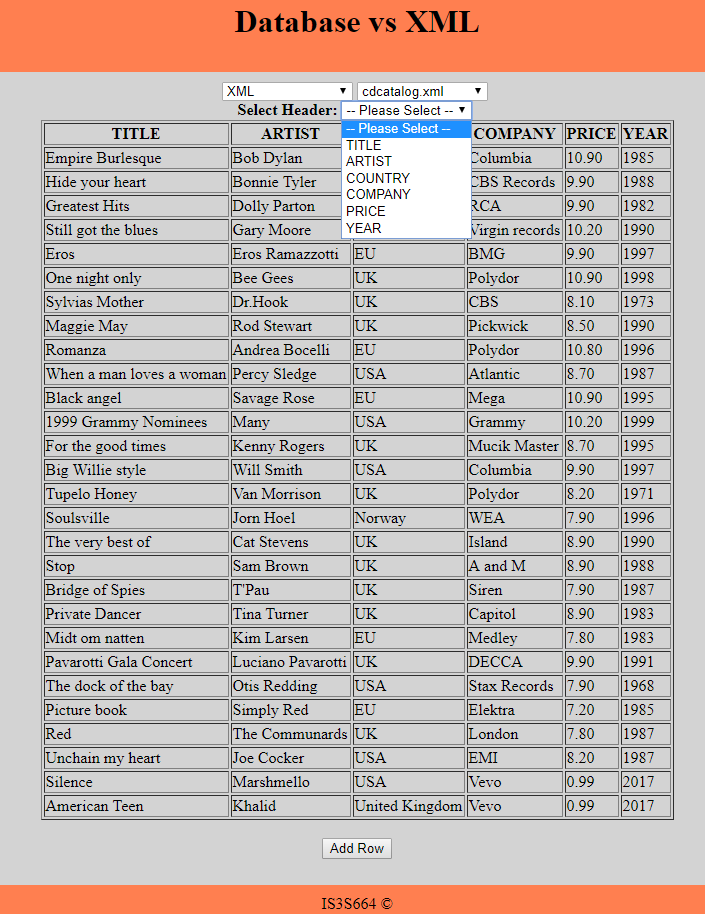
First of all, this isn’t really using any XML processing, but it is displaying all of the names of the files in the XML folder. This is done with the use of the glob function which is used to search a directory and in this case, select everything in that directory that has an .xml extension. It is then inserted into an array in the php script and returned to the index in JSON as you can see above, then they can be displayed as options in the select box as seen below.



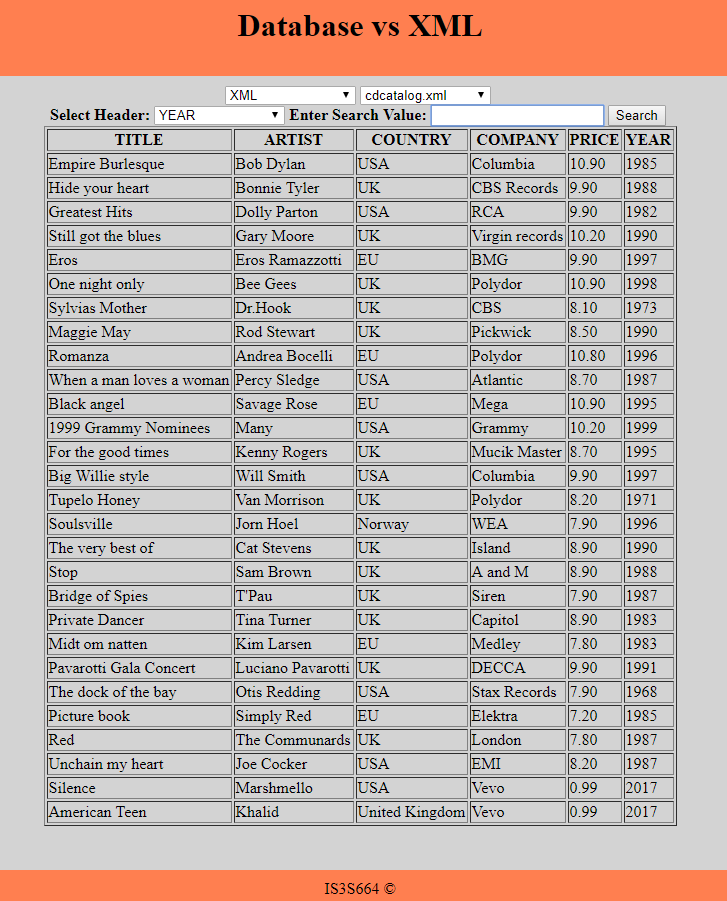
So, when a file is selected from the above drop down box this is the JSON that gets returned if for example it was cdcatalog.xml that was selected.



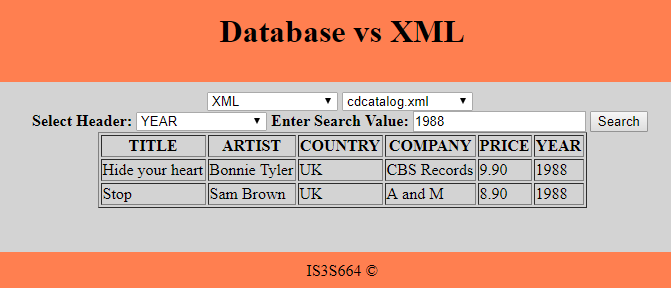
Throughout this system the simplexml load file function has been used to open the XML and XSD files for reading. So, to get the result above I first loaded the XSD file to get all of the child names, once that information is obtained you are able to load the XML file and loop through each child name and the values inserting them into an array. This can then be sent as JSON back to the home page to be displayed in the table that you see below.



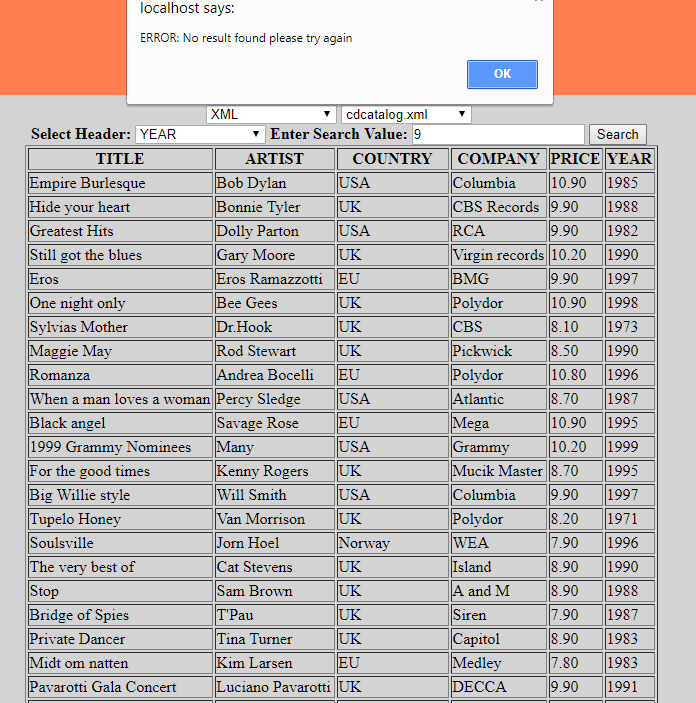
Above you can also see that there is a third drop down list that is filled by getting these child names from the XSD file again to be returned via JSON and inserted into the select box that you see displayed. Once one of these child names or column names is selected it gives the user the ability to search a certain value in that column as you can see below.

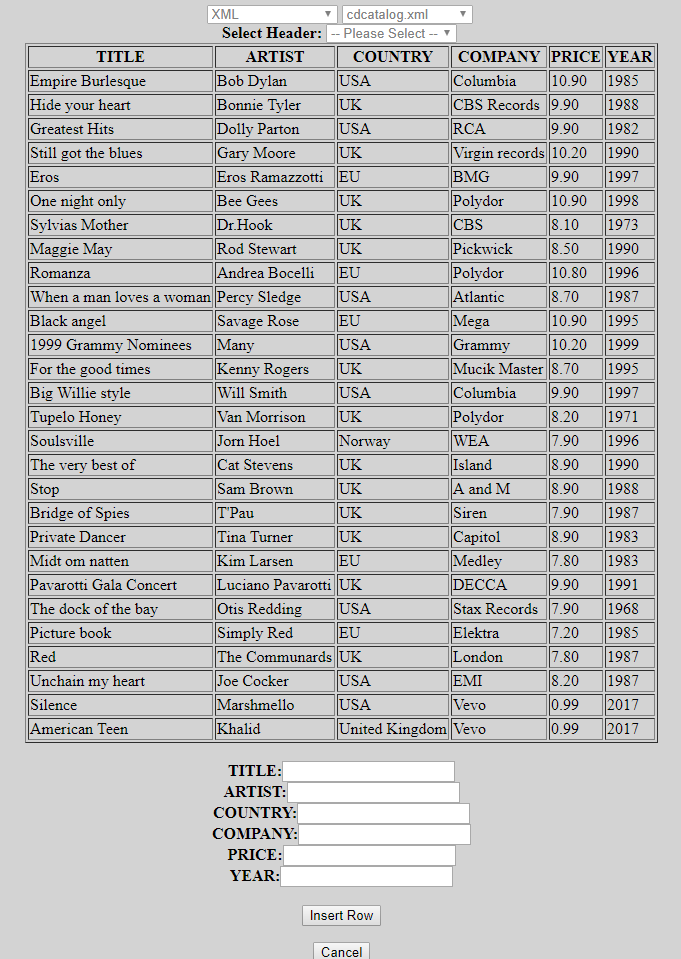


Above you can see the XML file cdcatalog is ready to be searched and below shows the search results. This is all done within a function that utilises an array of data in the file, the column name selected and the value searched. It then checks if the value exists anywhere in that column, if it does it then returns an array of all the results that contain that search value which you can see displayed below.

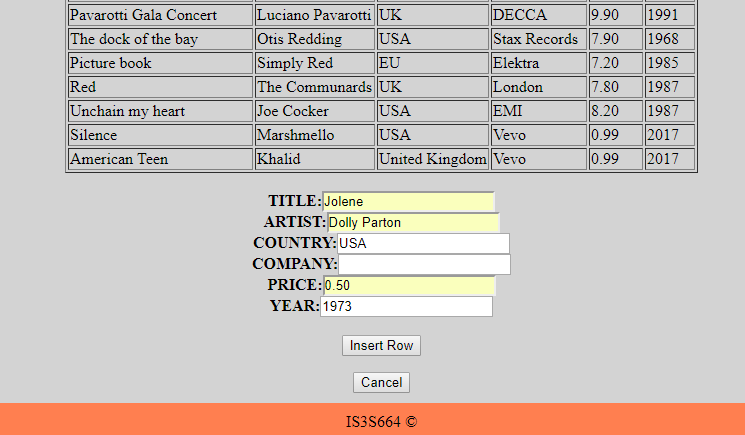


So, the only issue with the XML search is that you have to search for the exact value that’s in the file which is shown working above. Below it shows what happens if you just want to search for all rows that contain that value, it tells us that that field does not exist as the search does have be exact and it is also case sensitive. In the future this could be a feature that is worked on to be able to return all records that contain one or two letters or numbers that match the fields in that column.





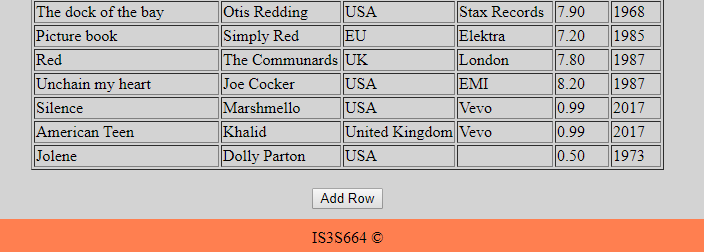
Above shows the form that is displayed when the add row button is pressed, it is created and filled with all the child names from the cdcatalog XML file. This is done in the same way as the column header drop down box by loading the schema for that file and returning all the child names. The form also has a hidden field that contains the name of the file/table itself, so that when the form gets submitted the php script knows which file/table it is dealing with.



Here you can see the data being inputted in to the form by the user and when the insert row button is pressed that data gets posted to the php script to be added to the XML file. This all is done once again by using simplexml functions and taking the values you see above and inserting them into the XML file that you see below.



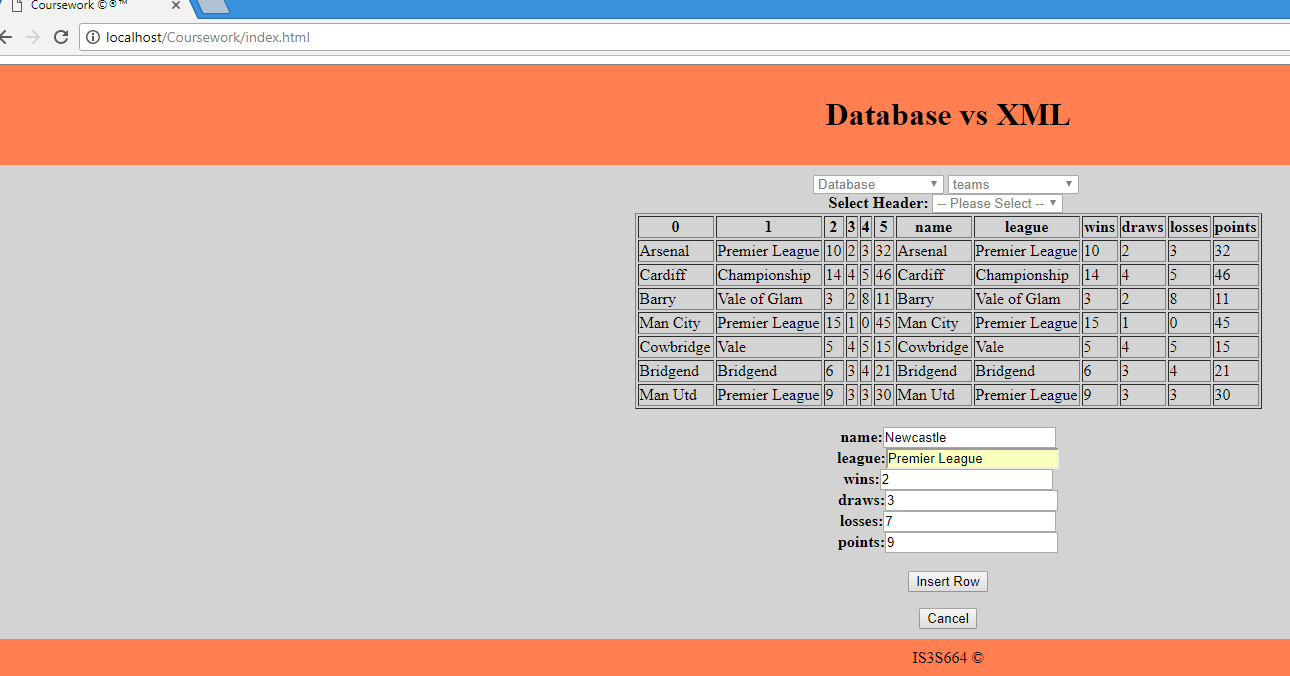
As you can see above it has added the data successfully, however it has been all inserted onto one line without any XML formatting. This works for now as it is still able to be read and displayed on the index.html as you can see below but in the future it is definitely something that I would consider tidying up so that when inserted it is done with proper formatting to match the rest of the XML document.



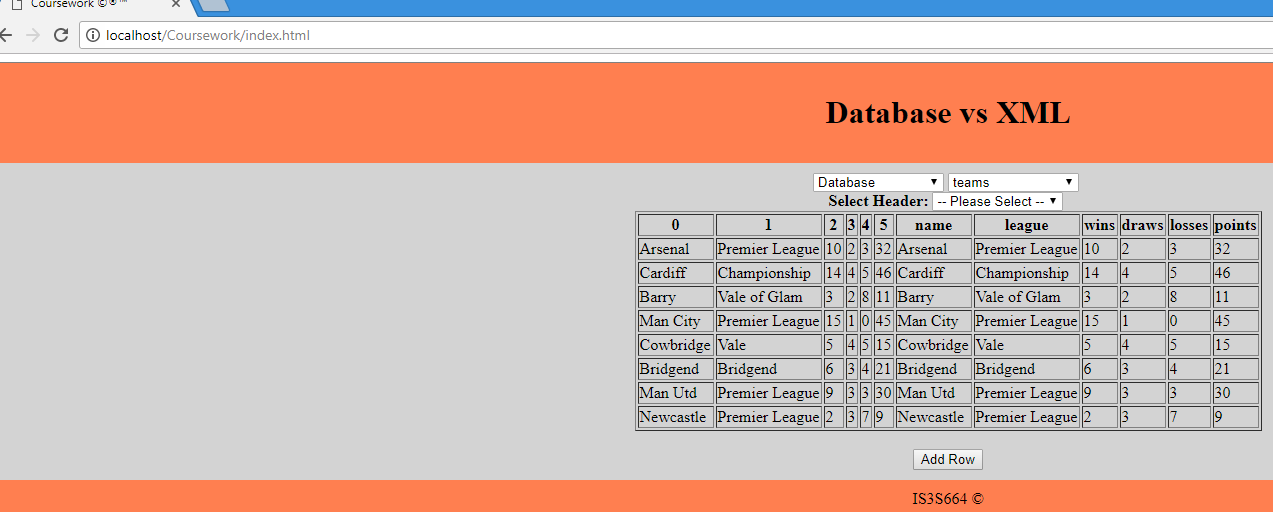
# **Testing different browsers**

**Chrome**

Displaying the form with the values ready for inserting

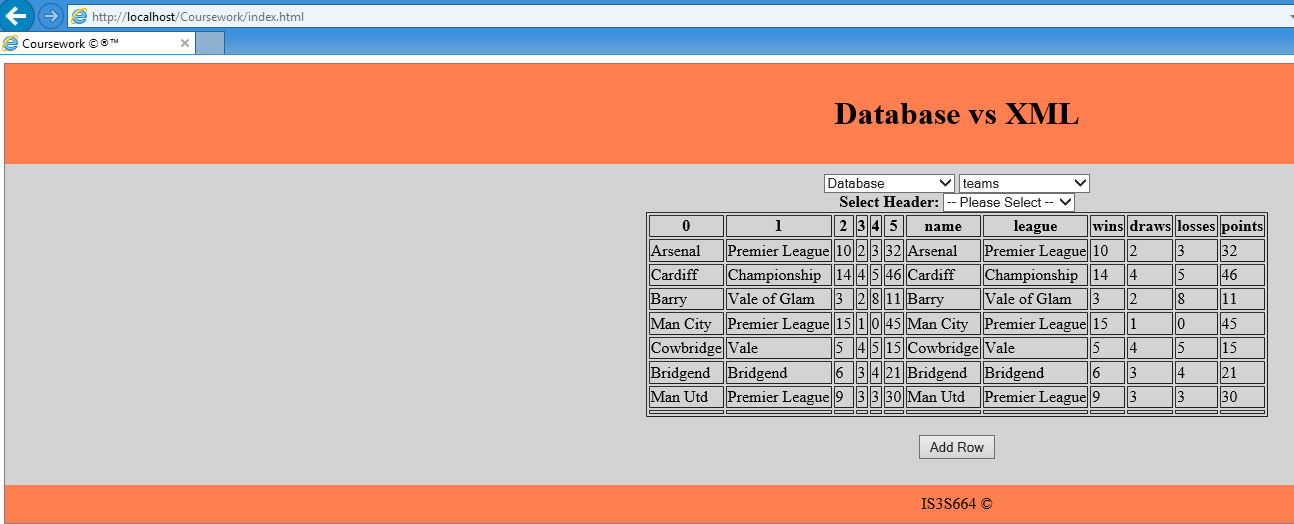


Displaying the new record in the teams table

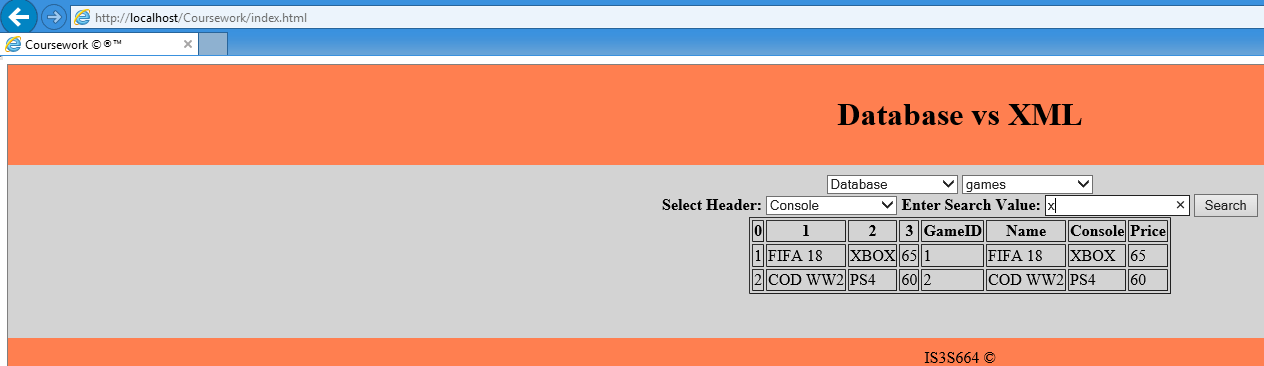


**Internet Explorer**

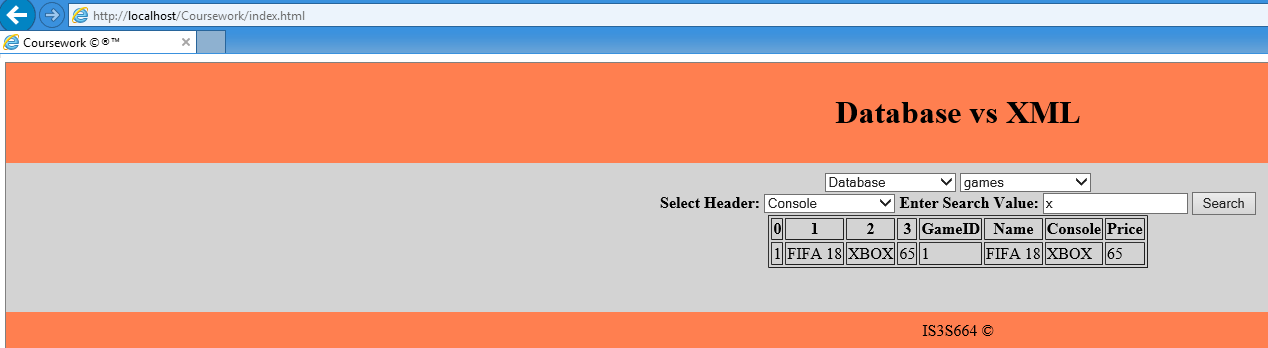
Displaying of database table teams



Entering a search value of ‘x’

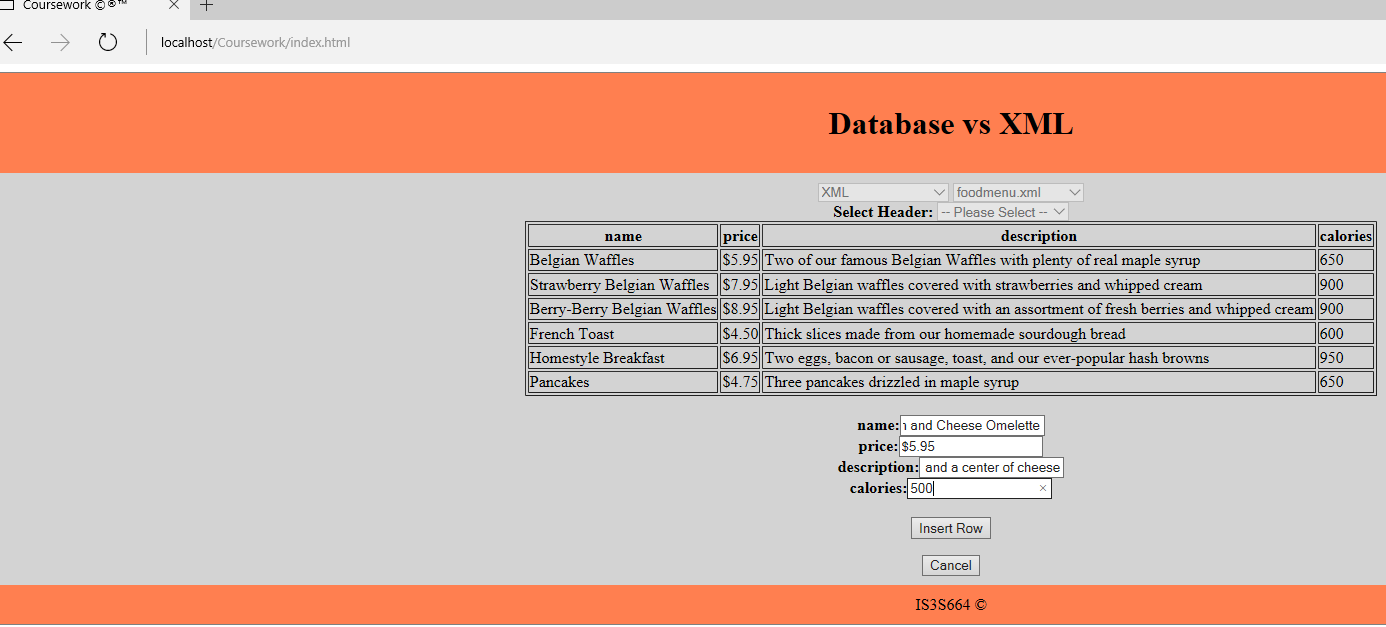


Showing the search results for ‘x’

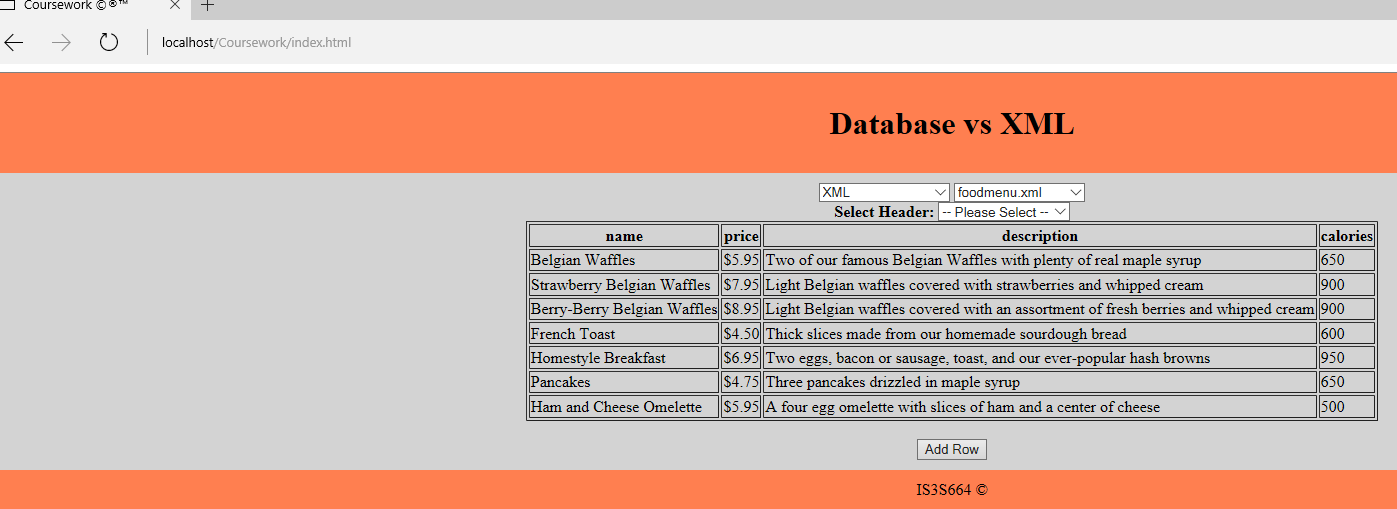


**Microsoft Edge**

Displaying the form with the values ready for inserting

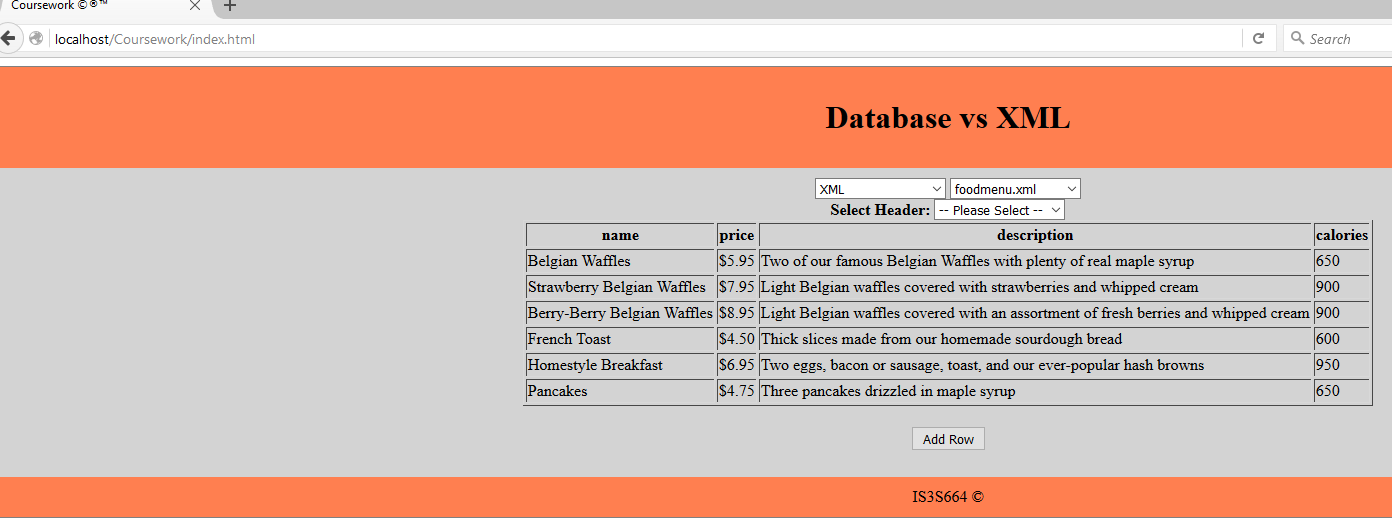


Displaying the new record in the teams table

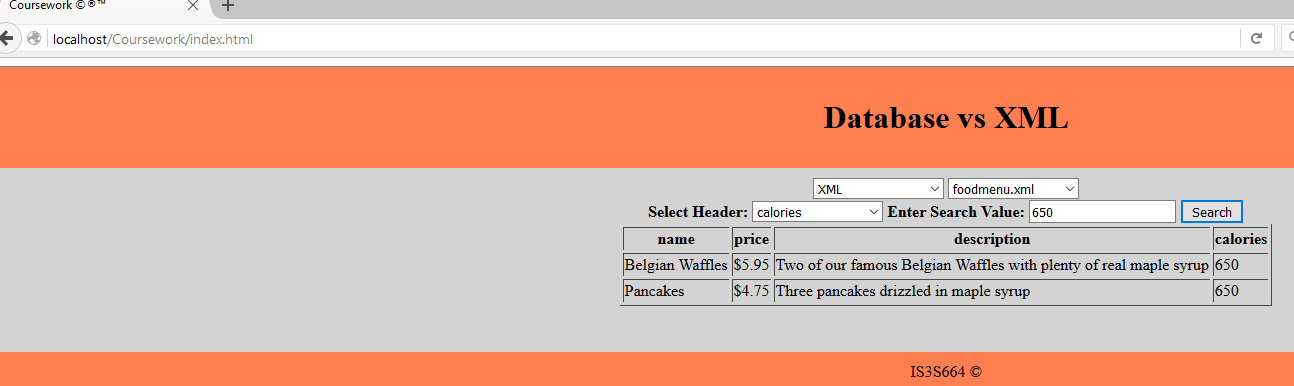


**Mozilla Firefox**

Displaying the XML file foodmenu



Showing the search results for ‘650’ calories



# **Code Listings**

**Index.html**

<html>

<head>

<title>Coursework &copy;&reg;&trade;</title>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

<script src="jquery.jCombo.js"></script>

<style type="text/css">

div.container {

width: 100%;

border: 1px solid gray;

}

table {

margin: auto;

}

header, footer {

padding: 10px;

color: black;

background-color: coral;

clear: left;

text-align: center;

}

article {

padding: 10px;

background-color: lightgrey;

text-align: center;

overflow: hidden;

}

label {

font-weight: bold;

}

.hide { position:absolute; top:-1px; left:-1px; width:1px; height:1px; }

</style>

</head>

<body>

<div class="container">

<header>

<h1>Database vs XML</h1>

</header>

<article>

<script type="text/javascript"> <!-- Start of Script -->

$(document).ready(function(){

<!-- hide the dropdown menu -->

$("#tablelist").hide();

$("#headlist").hide();

$("#searchbox").hide();

$("#searchbtn").hide();

$("#SelHead").hide();

$("#SearchVal").hide();

$("#addbtn").hide();

$("#cancbtn").hide();

///////////////////////////////////////////////////////////////////////////

///////////////////////////////Function////////////////////////////////////

///////////////////////////////////////////////////////////////////////////

function updateTable(){

setTimeout(function(){ //set the timer

if ($tablename.indexOf('xml') !== -1) { <!-- if the selected value is an xml doc type -->

$.getJSON("getXMLfile.php?filename="+$tablename, {}, <!-- Get the json data from this php file -->

function(json){

myList = json;

if(isEmpty(myList)) { //calls the function

alert("ERROR: No file was found");//when no results are found

} else {

$('#dvTable table').empty();

buildHtmlTable(); <!-- recieve the json and use it in this function to build the search results into a table -->

}

});

}

else <!-- if a database table is selected -->

{

$.getJSON("getDatabase.php?tablename="+$tablename, {}, <!-- Get the json data from this php file -->

function(json){

myList = json;

if(isEmpty(myList)) { //calls the function

alert("ERROR: No database was found"); //when no results are found

} else {

$('#dvTable table').empty();

buildHtmlTable(); <!-- recieve the json and use it in this function to build the search results into a table -->

}

});

}

}, 1000); //timer is 1 second

}

function isEmpty(myList) {

for(var key in myList) { <!-- This function checks if the object being sent to the page is empty or not-->

if(myList.hasOwnProperty(key))

return false;

}

return true;

}

var myList;

// Builds the HTML Table out of myList json data

function buildHtmlTable() {

var columns = addAllColumnHeaders(myList);

for (var i = 0 ; i < myList.length ; i++) {

var row$ = $('<tr/>');

for (var colIndex = 0 ; colIndex < columns.length ; colIndex++) {

var cellValue = myList[i][columns[colIndex]];

if (cellValue == null) { cellValue = ""; }

row$.append($('<td/>').html(cellValue));

}

$("#DataTable").append(row$);

}

}

// Adds a header row to the table and returns the set of columns.

function addAllColumnHeaders(myList)

{

var columnSet = [];

var headerTr$ = $('<tr/>');

for (var i = 0 ; i < myList.length ; i++) {

var rowHash = myList[i];

for (var key in rowHash) {

if ($.inArray(key, columnSet) == -1){

columnSet.push(key);

headerTr$.append($('<th/>').html(key));

}

}

}

$("#DataTable").append(headerTr$);

return columnSet;

}

var myForm

function buildForm(myForm){

$("#addDB").append("<input type='hidden' name='tablename' value='"+$tablename+"'>"); //add a hidden field conating the table name

for (var i = 0 ; i < myForm.length ; i++) {

$("#addDB").append("<label name='"+myForm[i]+"' >"+' '+myForm[i]+':'+"</label>"); //add a label

$("#addDB").append("<input type='text' name='"+myForm[i]+"'>"); //and add an input for each colum in the table

$("#addDB").append("<br>");

}

$("#addDB").append("<br>");

$("#addDB").append("<button type ='submit' id='inbtn'>Insert Row</button>"); //add a submit button to the form

$("#inbtn").click(function() //submit button is clicked

{

$("#addbtn").show(); //show the add button

$("#cancbtn").hide();

$("#addDB").hide(); //and hide all the form and buttons

$("#inbtn").hide();

document.getElementById("dblist").disabled = false;

document.getElementById("tablelist").disabled = false; <!-- enable all the selection boxes -->

document.getElementById("headlist").disabled = false;

updateTable(); //call the update table function to display the added record

});

}

/////////////////////////////////////////////////////////////////////////

/////////////////////////////Function////////////////////////////////////

/////////////////////////////////////////////////////////////////////////

$("#dblist").change(function() <!-- When the dropdown menu is changed -->

{

if($("#dblist").val() == 0) <!-- When "Please Select" is selected hide the menu -->

{

$("#tablelist").hide();

$("#headlist").hide();

$("#searchbox").hide();

$("#searchbtn").hide();

$("#SelHead").hide();

$("#SearchVal").hide();

$("#addbtn").hide();

$('#dvTable table').empty();

}

else if($("#dblist").val() == 1) <!-- when "XML" is selected display the XML dropdown menu -->

{

$("#tablelist").jCombo("getXMLlist.php", {} );

$("#tablelist").show();

}

else <!-- when "Database" is selected display the table dropdown menu -->

{

$("#tablelist").jCombo("getTablesList.php", {} );

$("#tablelist").show();

}

});

$("#tablelist").change(function() <!-- When the dropdown menu is changed -->

{

if($("#tablelist").val() == 0) <!-- When "Please Select" is selected hide the menu -->

{

$("#headlist").hide();

$("#searchbox").hide();

$("#searchbtn").hide();

$("#SelHead").hide();

$("#SearchVal").hide();

$("#addbtn").hide();

$('#dvTable table').empty();

}

else <!-- When anything else is selected -->

{

$("#SelHead").show();

$("#headlist").show();

$("#addbtn").show();

$tablename = $("#tablelist>option:selected").text(); <!-- put the text value of the option selected into a variable -->

if ($tablename.indexOf('xml') !== -1) { <!-- if the selected value is an xml doc type -->

$("#headlist").jCombo("getXMLhead.php?filename="+$tablename, {} ); <!-- fill the headlist dropdown box with data -->

$.getJSON("getXMLfile.php?filename="+$tablename, {}, <!-- Get the json data from this php file -->

function(json){

myList = json;

if(isEmpty(myList)) { //calls the function

alert("ERROR: No file was found");//when no results are found

} else {

$('#dvTable table').empty();

buildHtmlTable(); <!-- recieve the json and use it in this function to build the search results into a table -->

}

});

}

else <!-- if a database table is selected -->

{

$("#headlist").jCombo("getTheadList.php?tablename="+$tablename, {} ); <!-- fill the headlist dropdown box with data -->

$.getJSON("getDatabase.php?tablename="+$tablename, {}, <!-- Get the json data from this php file -->

function(json){

myList = json;

if(isEmpty(myList)) { //calls the function

alert("ERROR: No database was found"); //when no results are found

} else {

$('#dvTable table').empty();

buildHtmlTable(); <!-- recieve the json and use it in this function to build the search results into a table -->

}

});

}

}

});

$("#headlist").change(function() <!-- When the dropdown menu is changed -->

{

if($("#headlist").val() == 0) <!-- When "Please Select" is selected hide the textbox and button-->

{

$("#searchbox").hide();

$("#searchbtn").hide();

$("#SearchVal").hide();

$("#addbtn").show();

document.getElementById("searchbox").value = ""; //clears the searchbox

if ($tablename.indexOf('xml') !== -1) { <!-- if the selected value is an xml doc type -->

$.getJSON("getXMLfile.php?filename="+$tablename, {}, <!-- Get the json data from this php file -->

function(json){

myList = json;

if(isEmpty(myList)) { //calls the function

alert("ERROR: No file was found");//when no results are found

} else {

$('#dvTable table').empty();

buildHtmlTable(); <!-- recieve the json and use it in this function to build the search results into a table -->

}

});

}

else <!-- if a database table is selected -->

{

$.getJSON("getDatabase.php?tablename="+$tablename, {}, <!-- Get the json data from this php file -->

function(json){

myList = json;

if(isEmpty(myList)) { //calls the function

alert("ERROR: No database was found"); //when no results are found

} else {

$('#dvTable table').empty();

buildHtmlTable(); <!-- recieve the json and use it in this function to build the search results into a table -->

}

});

}

}

else <!-- When anything else is selected -->

{

$("#searchbox").show();

$("#searchbtn").show();

$("#SearchVal").show();

$("#addbtn").hide();

$("#searchbox").focus();

$headname = $("#headlist>option:selected").text(); <!-- put the text value of the option selected into a variable -->

}

});

$("#searchbtn").click(function()

{

$searchval = $("#searchbox").val(); <!-- put the value of the textbox into a variable -->

if ($searchval == "")

{

alert("ERROR: please enter a value in the text box before searching");

}

else

{

if ($tablename.indexOf('xml') !== -1) <!-- if the selected value is an xml doc type -->

{

$.getJSON("getXMLsearch.php", <!-- get the json from the php file -->

{

filename: $tablename,

headname: $headname, <!-- send these variables to the php file -->

searchval: $searchval

},

function(json){

myList = json;

if(isEmpty(myList)) { //calls the function

alert("ERROR: No result found please try again"); //when no results are found

} else {

$('#dvTable table').empty();

buildHtmlTable(); <!-- recieve the json and use it in this function to build the search results into a table -->

}

});

}

else <!-- if a database table is selected -->

{

$.getJSON("getSearch.php", <!-- get the json from the php file -->

{

tablename: $tablename,

headname: $headname, <!-- send these variables to the php file -->

searchval: $searchval

},

function(json){

myList = json;

if(isEmpty(myList)) { //calls the function

alert("ERROR: No result found please try again"); //when no results are found

} else {

$('#dvTable table').empty();

buildHtmlTable(); <!-- recieve the json and use it in this function to build the search results into a table -->

}

});

}

}

});

$("#addbtn").click(function()

{

$("#addbtn").hide();

$("#cancbtn").show();

$("#addDB").show();

document.getElementById("dblist").disabled = true;

document.getElementById("tablelist").disabled = true; <!-- disable all the selection boxes while adding rows -->

document.getElementById("headlist").disabled = true;

if ($tablename.indexOf('xml') !== -1) { <!-- if the selected value is an xml doc type -->

$.getJSON("getXMLadd.php?filename="+$tablename, {}, <!-- Get the json data from this php file -->

function(json){

myForm = json;

$('#dvTable form').empty(); //empty the form

document.addNewRow.action = "addXMLfile.php"; //set the form action to be the add xml php file

buildForm(myForm); //call this function to build the form

});

}

else { <!-- When anything else is selected -->

$.getJSON("getdbadd.php?filename="+$tablename, {}, <!-- Get the json data from this php file -->

function(json){

myForm = json;

$('#dvTable form').empty(); //empty the form

document.addNewRow.action = "addDB.php"; //set the form action to be the add db php file

buildForm(myForm); //call this function to build the form

});

}

});

$("#cancbtn").click(function()

{

$("#addbtn").show();

$("#cancbtn").hide();

$('#dvTable form').empty();

document.getElementById("dblist").disabled = false;

document.getElementById("tablelist").disabled = false; <!-- enable all the selection boxes -->

document.getElementById("headlist").disabled = false;

});

$('#searchbox').keypress(function(e){

if(e.keyCode==13) <!-- if enter is pressed -->

$('#searchbtn').click();

});

});

</script> <!-- End of Script -->

<!-- create the drop down menu that has options of "XML" or "Database" -->

<select name="dblist" id="dblist">

<option value="0">-- Please Select --</option>

<option value="1">XML</option>

<option value="2">Database</option>

</select>

<!-- create the drop down menu that appears when "Database" is selected -->

<select name="tablelist" id="tablelist"></select>

<br>

<label id="SelHead">Select Header: </label>

<select name="headist" id="headlist"></select>

<label id="SearchVal"> Enter Search Value: </label>

<input type ="text" name="search" id="searchbox">

<button type ="button" id="searchbtn">Search</button>

<br>

<!-- create a div called dvTable -->

<div id="dvTable" class="form">

<!-- create the table that will be filled and appear when a data source has been selected -->

<table id="DataTable" border="1">

</table>

<br>

<iframe name="hiddenFrame" class="hide"></iframe>

<button type ="button" id="addbtn">Add Row</button>

<!-- create the form that will be filled and appear when the add row button is pressed -->

<form id="addDB" name="addNewRow" method="post" target="hiddenFrame">

</form>

<button type ="button" id="cancbtn">Cancel</button>

</div>

</article>

<footer>IS3S664 &copy;</footer>

</div>

</body>

</html>

**getXMLlist.php**

<?php

$directory = 'XML'; //set the variable to be the name of the folder where the xml files are saved

if (! is\_dir($directory)) { //if this directory doesnt exist

exit('Invalid diretory path'); //display error

}

$files = array(); //create array

$i=0; //create i and set it to 0

foreach (glob("XML\\*.xml") as $file) { //for each .xml file found in the directory

$file = trim($file,"XML\\"); //trim the directory XML\ from the file variable

$i++; //add 1 to i

$files[] = array (strval($i), $file); //add i and file name to the array

}

echo json\_encode($files); //echo the array in a json format

?>

**getXMLfile.php**

<?php

$filename = $\_GET["filename"]; //get filename from index.html into variable

$childname;

$lines = array(); //create array

$xsdname = trim($filename,".xml"); //trim the .xml off the filename and set it to a different variable

$ext = ".xsd";

$xsdname = $xsdname . $ext; //put the variables together to create a new filename

$xsd=simplexml\_load\_file("XML\\".$xsdname) or die("Error: Cannot create object"); //load the schema from the folder, if fails display error

$elementName = array\_map('strval', $xsd->xpath('//xs:element/@name')); //put the element names into a variable

$items = array(); //create arry

foreach ($elementName as $value)

{

$items[] = array ($value); //loop and fill the array with the element names

}

$childname = array\_shift( $items ); //set variable to the first value of the array

$childname = serialize($childname); //serializes the array object to a string

$childname = substr($childname, 14); //truncates the first 14 characters from the variable

$childname = trim($childname,'";}[]'); //trims that part of the string to get the childname

$xml=simplexml\_load\_file("XML\\".$filename) or die("Error: Cannot create object");//load the xml file from the folder, if fails display error

foreach($xml->$childname as $child)

{

$line = array();//create array

foreach($child as $key => $value) //loop through all the child elements

{

$line[(string)$key]=(string)$value;

}

$lines[] = $line; //insert into array

}

echo json\_encode($lines); //sends the json back to the index.html

?>

**getXMLhead.php**

<?php

$filename = $\_GET["filename"]; //get filename from index.html into variable

$filename = trim($filename,".xml"); //trim the .xml off the filename and set it to a different variable

$ext = ".xsd";

$filename = $filename . $ext; //put the variables together to create a new filename

$xml=simplexml\_load\_file("XML\\".$filename) or die("Error: Cannot create object"); //load the schema from the folder, if fails display error

$elementName = array\_map('strval', $xml->xpath('//xs:element[not(node())]/@name')); //put the element names into a variable

$items = array(); //create array

$i=0; //set i to 0

foreach ($elementName as $value)

{

$i++; //add 1 to i

$items[] = array (strval($i), $value); //loop and fill the array with i and the element names

}

echo json\_encode($items); //output the json back to the index.html

?>

**getXMLsearch.php**

<?php

$filename = $\_GET["filename"]; //get filename from index.html into variable

$childname;

$headname = $\_GET["headname"]; //get column to search from index.html into variable

$searchval = $\_GET["searchval"]; //get what to search from index.html into variable

$lines = array(); //create array

$xsdname = trim($filename,".xml"); //trim the .xml off the filename and set it to a different variable

$ext = ".xsd";

$xsdname = $xsdname . $ext; //put the variables together to create a new filename

$xsd=simplexml\_load\_file("XML\\".$xsdname) or die("Error: Cannot create object"); //load the schema from the folder, if fails display error

$elementName = array\_map('strval', $xsd->xpath('//xs:element/@name')); //put the element names into a variable

$items = array(); //create arry

foreach ($elementName as $value)

{

$items[] = array ($value); //loop and fill the array with the element names

}

$childname = array\_shift( $items ); //set variable to the first value of the array

$childname = serialize($childname); //serializes the array object to a string

$childname = substr($childname, 14); //truncates the first 14 characters from the variable

$childname = trim($childname,'";}[]'); //trims that part of the string to get the childname

$xml=simplexml\_load\_file("XML\\".$filename) or die("Error: Cannot create object"); //load the xml file from the folder, if fails display error

foreach($xml->$childname as $child)

{

$line = array();//create array

foreach($child as $key => $value) //loop through all the child elements

{

$line[(string)$key]=(string)$value;

}

$lines[] = $line; //insert into array

}

function search($array, $key, $value) //create function search and get variables

{

$results = array();

if (is\_array($array)) {

if (isset($array[$key]) && $array[$key] == $value) {

$results[] = $array;

}

foreach ($array as $subarray) {

$results = array\_merge($results, search($subarray, $key, $value));

}

}

return $results;

}

echo json\_encode(search($lines, $headname, $searchval)); //call function and send json back to index.html

?>

**getXMLadd.php**

<?php

$filename = $\_GET["filename"]; //get filename from index.html into variable

$filename = trim($filename,".xml"); //trim the .xml off the filename and set it to a different variable

$ext = ".xsd";

$filename = $filename . $ext; //put the variables together to create a new filename

$xml=simplexml\_load\_file("XML\\".$filename) or die("Error: Cannot create object"); //load the schema from the folder, if fails display error

$elementName = array\_map('strval', $xml->xpath('//xs:element[not(node())]/@name')); //put the element names into a variable

$items = array(); //create array

foreach ($elementName as $value)

{

$items[] = array ($value); //loop and fill the array with element names

}

echo json\_encode($items); //output the json back to the index.html

?>

**addXMLfile.php**

<?php

foreach ($\_POST as $key => $value) { //loop through all values to find the file name

if ($key == "tablename"){ //if found

$filename = $value; //put it into the variable

}

}

$childname;

$lines = array(); //create array

$xsdname = trim($filename,".xml"); //trim the .xml off the filename and set it to a different variable

$ext = ".xsd";

$xsdname = $xsdname . $ext; //put the variables together to create a new filename

$xsd=simplexml\_load\_file("XML\\".$xsdname) or die("Error: Cannot create object"); //load the schema from the folder, if fails display error

$elementName = array\_map('strval', $xsd->xpath('//xs:element/@name')); //put the element names into a variable

$items = array(); //create arry

foreach ($elementName as $value)

{

$items[] = array ($value); //loop and fill the array with the element names

}

$childname = array\_shift( $items ); //set variable to the first value of the array

$childname = serialize($childname); //serializes the array object to a string

$childname = substr($childname, 14); //truncates the first 14 characters from the variable

$childname = trim($childname,'";}[]'); //trims that part of the string to get the childname

$folder = "XML\\";

$filename = $folder . $filename; //put these variables together to create the file directory

$xml=simplexml\_load\_file($filename) or die("Error: Cannot create object");//load the xml file from the folder, if fails display error

$row = $xml->addChild($childname);

foreach ($\_POST as $key => $value) { //loop through all the keys and values

if ($key == "tablename"){ //if table name dont do anything

}

else{ //if anything else

$row->addChild($key, $value); //insert key and value

}

}

$xml->asXML($filename); //insert the values as XML into the xml file

?>

**getTablesList.php**

<?php

include "mysqlconnect.php";

$ok=true;

try {

$dbh = new PDO('mysql:host='.$hostname.';dbname='.$database, $username, $password); // connect to database

}

catch (PDOException $e) { //if failed connection

$ok = false; //set ok to be false

}

if($ok) //if ok is true

{

try {

$tableschema = $database; //set the tableschema to be the name of the database

$sql = $dbh->prepare('SELECT TABLE\_NAME FROM information\_schema.TABLES WHERE TABLE\_SCHEMA = :tableschema'); //create the sql statement

$sql->bindParam(':tableschema', $tableschema); //put the value of tableschema into the sql

$sql->execute(); //execute the sql statement

if ( $sql->columnCount() > 0 ) //if data was found

{

$items = array(); //create array

$i=0; //create i and set it to 0

while($row=$sql->fetch()) //while there is rows in the database

{

$i++; //add 1 to i

$items[] = array (strval($i), $row[0]); //add the value of i and the rows into the array

}

echo json\_encode($items); //echo the array in a json format

}

else //if no data found

echo '{"error":"Database lookup failed"}'; //display error

}

catch (PDOException $e) {

$ok = false;

echo '{"error":"Database lookup failed"}'; //display error

}

}

else //if ok was false

echo '{"error":"Database connection failed"}'; //display error

?>

**getDatabase.php**

<?php

include "mysqlconnect.php";

$ok=true;

try {

$dbh = new PDO('mysql:host='.$hostname.';dbname='.$database, $username, $password); //connect to database

}

catch (PDOException $e) { //if failed oonnection

$ok = false; //set ok to be false

}

if($ok) //if ok is true

{

try {

$tablename = $\_GET["tablename"]; //gets the table name selected from the index.html into a variable

$sql = $dbh->prepare('SELECT \* FROM '. $tablename .''); //create the sql statement

$sql->execute(); //execute the sql statement

if ( $sql->columnCount() > 0 ) //if data was found

{

$items = array(); //create array

while($row=$sql->fetch()) //while there is rows in the database

{

$items[] = $row; //add the rows to the array

}

echo json\_encode($items); //echo the array in a json format

}

else //if no data found

echo '{"error":"Database lookup failed"}'; //display error

}

catch (PDOException $e) {

$ok = false;

echo '{"error":"Database lookup failed"}'; //display error

}

}

else //if ok was false

echo '{"error":"Database connection failed"}'; //display error

?>

**getTheadList.php**

<?php

include "mysqlconnect.php";

$ok=true;

try {

$dbh = new PDO('mysql:host='.$hostname.';dbname='.$database, $username, $password); //connect to database

}

catch (PDOException $e) { //if failed connection

$ok = false; //set ok to be false

}

if($ok) //if ok is true

{

try {

$tableschema = $database; //set the tableschema to be the name of the database

$tablename = $\_GET["tablename"]; //get the name of the table from index.html into the variable

$sql = $dbh->prepare('SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_SCHEMA = :tableschema AND TABLE\_NAME = :tablename'); //create the sql statement

$sql->bindParam(':tableschema', $tableschema);

$sql->bindParam(':tablename', $tablename); //put the values of the variables into the sql

$sql->execute(); //execute the sql statemtn

if ( $sql->columnCount() > 0 ) //if data was found

{

$items = array(); //create array

$i=0; //create i and set it to 0

while($row=$sql->fetch()) // while there is rows in the database

{

$i++; //add 1 to i

$items[] = array (strval($i), $row[0]); //add the value of i and the rows into the array

}

echo json\_encode($items); //echo the array in a json format

}

else //if no data found

echo '{"error":"Database lookup failed"}'; //display error

}

catch (PDOException $e) {

$ok = false;

echo '{"error":"Database lookup failed"}'; //display error

}

}

else //if ok was false

echo '{"error":"Database connection failed"}'; //display error

?>

**getSearch.php**

<?php

include "mysqlconnect.php";

$ok=true;

try {

$dbh = new PDO('mysql:host='.$hostname.';dbname='.$database, $username, $password); //connect to database

}

catch (PDOException $e) { //if failed connection

$ok = false; //set ok to be false

}

if($ok) //if ok is true

{

try {

$tablename = $\_GET["tablename"];

$headname = $\_GET["headname"]; //gets the values from the index.html into the variables

$searchval = $\_GET["searchval"];

$sql = $dbh->prepare("SELECT \* FROM $tablename WHERE $headname LIKE '%$searchval%'"); //create the sql statement

$sql->execute(); //execute the sql statement

if ( $sql->columnCount() > 0 ) //if data was found

{

$items = array(); //create array

while($row=$sql->fetch()) //while there is rows in the database

{

$items[] = $row; //add the rows to the array

}

echo json\_encode($items); //echo the array in a json format

}

else //if no data found

echo '{"error":"Database lookup failed"}'; //display error

}

catch (PDOException $e) {

$ok = false;

echo '{"error":"Database lookup failed"}'; //display error

}

}

else //if ok was false

echo '{"error":"Database connection failed"}'; //display error

?>

**getdbadd.php**

<?php

include "mysqlconnect.php";

$ok=true;

try {

$dbh = new PDO('mysql:host='.$hostname.';dbname='.$database, $username, $password); //connect to database

}

catch (PDOException $e) { //if failed connection

$ok = false; //set ok to be false

}

if($ok) //if ok is true

{

try {

$tableschema = $database; //set the tableschema to be the name of the database

$tablename = $\_GET["filename"]; //get the name of the table from index.html into the variable

$sql = $dbh->prepare('SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_SCHEMA = :tableschema AND TABLE\_NAME = :tablename'); //create the sql statement

$sql->bindParam(':tableschema', $tableschema);

$sql->bindParam(':tablename', $tablename); //put the values of the variables into the sql

$sql->execute(); //execute the sql statemtn

if ( $sql->columnCount() > 0 ) //if data was found

{

$items = array(); //create array

while($row=$sql->fetch()) // while there is rows in the database

{

$items[] = array ($row[0]); //add the value of i and the rows into the array

}

echo json\_encode($items); //echo the array in a json format

}

else //if no data found

echo '{"error":"Database lookup failed"}'; //display error

}

catch (PDOException $e) {

$ok = false;

echo '{"error":"Database lookup failed"}'; //display error

}

}

else //if ok was false

echo '{"error":"Database connection failed"}'; //display error

?>

**addDB.php**

<?php

include "mysqlconnect.php";

$ok=true;

try {

$dbh = new PDO('mysql:host='.$hostname.';dbname='.$database, $username, $password); //connect to database

}

catch (PDOException $e) { //if failed connection

$ok = false; //set ok to be false

}

if($ok) //if ok is true

{

try {

$fields = array();

$data = array();

$tablename;

foreach ($\_POST as $key => $value) {

if($key == "tablename") {

$tablename = $value;

} //this loops sorts the posted values into three different variables

else{

$fields[] = $key;

$data[] = $value;

}

}

$sql = ('INSERT INTO teams(name, league, wins, draws, losses, points) VALUES(?,?,?,?,?,?)'); //creation of sql insert statement

$stmt = $dbh->prepare($sql); //prepares the sql statement

$stmt ->execute($data); //executes the sql statement putting the array of data into the values to be inserted

}

catch (PDOException $e) {

$ok = false;

echo '{"error":"Database lookup failed"}'; //display error

}

}

else //if ok was false

echo '{"error":"Database connection failed"}'; //display error

?>