

Based on 100 points total, roughly 1.1 per check box unless noted as a %. Extra credit evaluated separately.

Lab 2 - Display records.

The purpose of this exercise is to:

Create a database, use a constants file and a class file to connect to your database and display the records on the index page. You will also need to use version control software. If that is not working you cannot receive a grade higher than 50% even if you have completed more than half of the check boxes.

As a reminder all labs are due by noon. Many classes have them due by midnight but I have extended all the due dates to noon the next day to allow you a little more time.

Set up version control for this lab.

- ☐ Make a copy of your dev-lab1 folder naming it dev-lab2. i.e. just copy the whole folder and rename the copy as this will have all the files to start.
- ☐ Be sure to have a README.md with just your First and Last Name.
- ☐ Create a new private GitHub project in the class Repo named Lab-2-FirstName-LastName. For example mine would be Lab-2-Robert-Erickson.
- ☐ Verify that you have used the case and hypens in your repo name as detailed above.
- ☐ Keep the content of the first GitHub page saving them in gitcommands.txt overwriting the commands from lab 1. This file should contain only the commands for lab2.
- ☐ Erase all previous commits from your dev-lab2 folder so we can start new. Be sure you are in the correct folder dev-lab2

```
rm -rf .git
```

- ☐ Set up your dev-lab2 folder to be a git repo (git init) on your computer.

```
git init
```

- ☐ Check if a remote is connected and remove it. It should already be deleted but it does not hurt to check.

```
git remote -v  
git remote rm origin
```

- ☐ Now connect your local computer to your git repo:

```
git remote add origin command
```

- ☐ Verify you still have the file: .user.ini in your dev-lab2 folder.
- ☐ Verify you still have the file named .gitignore
- ☐ add pass.php to your .gitignore file (you have not made the file yet but we want to ignore it when you do).
- ☐ Make a commit after you sFTP your files and they are working as before.

Set up the live site.

- ☐ Using Terminal or git bash shell, connect to the server and get into your class folder creating the live folder (live-lab2)
 - ☐ Initialize live-lab2 as a git repo.
 - ☐ Connect your live folder with the GitHub repo.
 - ☐ Pull your files live.
 - ☐ After you pull always verify the site is working.
-

Create a database.

If you do not have a **mySQL account** you will need to create one. [Request a mySQL username and password](#) . Be sure to save the auto generated email.


- ☐ Create a new database naming it cs148_lab 2. <https://webdb.uvm.edu/account/> NOTE the actual name will be YOURNETID_cs148_lab 2

Create a database

Create a new database named RERICKSO_ .

At the bottom of the Account page is where you create a new database.

- ☐ Using phpMyAdmin (sql or web form) create a table named tblWildlife based on this data dictionary.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	pmkWildlifeId 	int(11)			No	None		AUTO_INCREMENT
2	fldType	varchar(12)	utf8_general_ci		No	None		
3	fldCommonName	varchar(20)	utf8_general_ci		No	None		
4	fldDescription	varchar(900)	utf8_general_ci		No	None		
5	fldHabitat	text	utf8_general_ci		No	None		
6	fldReproduction	text	utf8_general_ci		No	None		
7	fldDiet	text	utf8_general_ci		No	None		
8	fldManagement	text	utf8_general_ci		No	None		
9	fldStatus	text	utf8_general_ci		No	None		
10	fldMainImage	varchar(30)	utf8_general_ci		No	None		

Data Dictionary.

- ☐ Save the create table sql commands in sql.php. If you used the form, export the table and view the sql to get the create table statement.

```

<?php
include 'top.php';
?>
<main>
    <p>Create Table SQL</p>

    <pre>
CREATE TABLE tblIssues(
    pmkIssuesId INT AUTO_INCREMENT PRIMARY KEY,
    fldActivity VARCHAR(40),
    fldEffects VARCHAR(200),
    fldImpact VARCHAR(200)
)
    </pre>

</main>
<?php include 'footer.php'; ?>
</body>
</html>

```

sample sql.php (note you will have select query not a create query)

Populate the database table.

- ☐ Choose any three animals (mammals) from the [Vt Fish and Wildlife](#) site.
- ☐ Save the image of the animals you chose in dev-lab 2/images.
- ☐ Using phpMyAdmin insert the information for the three animals that you chose.

- ☐ Browse the table and take a screen shot of your records naming it wildlife-records.png
-

PHP Constant file.

Constants allow us to set properties for your program. We are going to start with seven properties.

- ☐ Create a file named constants.php in your dev-lab 2/lib folder and define the constants below in this file.
- ☐ Define a constant flag to debug your code. Name this constant DEBUG
- ☐ We need to use several server variables. So before we use them (they come from the client), sanitize the super global server array first.
- ☐ Create a constant named SERVER that has the sanitized \$_SERVER['SERVER_NAME'] in it.
- ☐ Create a constant named DOMAIN that add two slashes (//) and the SERVER constant you just created.
- ☐ Create a constant named PHP_SELF that has the sanitized server variable PHP_SELF
- ☐ Create a constant named PATH_PARTS that has the array pathinfo(PHP_SELF) in it NOTE: this is used in top.php and nav.php already to pull out the filename.
- ☐ Create a constant named BASE_PATH that is the DOMAIN plus the dirname with a slash (/)
We use this in code to identify the absolute path structure.
- ☐ Create a constant named LIB_PATH that has the relative path to the lib folder.
- ☐ Add a debug statement to print out the values of DOMAIN, PHP_SELF, the PATH_PARTS array, BASE_PATH and LIB_PATH.
- ☐ In top.php include the constants file after you link your style sheets.

```
15  <!-- *** include libraries *** -->
16  <?php
17  include 'lib/constants.php';
18
```

Include the constants.


- ☐ sFTP your files to your dev folder.
- ☐ Verify that each file has the active class printing in the nav for that page.
- ☐ Verify that each file has the filename printing in the body id attribute for that page.
- ☐ Make a commit for constants.php

Your file should be similar to my sample [constants.php](#)

Set up a class file to connect to your database

pass.php

- ☐ Create a file named pass.php in your lib folder with your reader and writer passwords.

```
lib >  pass.php
1  <?php
2  $dbReader="xxxx";
3  $dbWriter="xxxx";
4  ?>
```


variable names for your passwords

No need to make a commit for pass.php, as it is an ignored file.

- ☐ Make sure you ftp lib/pass.php to the live site manually since it is ignored by git.

Database.php

- ☐ Create a file named Database.php in your lib folder
- ☐ Add a pdo variable and debug constant like this:

```
lib >  Database.php
1  <?php
2  class DataBase{
3      public $pdo = '';
4
5      const DB_DEBUG = false;
6
7  } // ends the class
```

Step one for your class file Define variables and constants

- ☐ Add a class constructor just before the closing bracket. Note there are two underscores before construct __ vs just one _

```
5      const DB_DEBUG = false;
6
7      public function __construct($dataBaseUser, $whichDataBasePassword, $dataBaseName) {
8          $this->pdo = null;
9
10         // passwords be sure to add pass.php to .gitignore
11         include 'pass.php';
12         $DataBasePassword = '';
13
14         switch ($whichDataBasePassword) {
15             case 'r':
16                 $DataBasePassword = $dbReader;
17                 break;
18             case 'w':
19                 $DataBasePassword = $dbWriter;
20                 break;
21         }
22
23         $query = NULL;
24
25         $dsn = 'mysql:host=webdb.uvm.edu;dbname=';
```

Step two for your class creating the constructor part one

- ☐ Add a little debugging code for your class constructor.

```
27         if (self::DB_DEBUG){
28             echo "<p>Try connecting with phpMyAdmin with these credentials.</p>";
29             echo '<p>Username: ' . $dataBaseUser;
30             echo '<p>DSN: ' . $dsn . $dataBaseName;
31             echo '<p>Password: ' . $DataBasePassword;
32         }
```

Step three for your class adding a little debugging

- ☐ Add the code to make your connection with a try catch statement.

```
34         try {
35             $this->pdo = new PDO($dsn . $dataBaseName, $dataBaseUser, $DataBasePassword)
36             ;
37
38             if (!$this->pdo) {
39                 if (self::DB_DEBUG) echo '<p>You are NOT connected to the database!</p>';
40                 return 0;
41             } else {
42                 if (self::DB_DEBUG) echo '<p>You are connected to the database!</p>';
43                 return $this->pdo;
44             }
45         } catch (PDOException $e) {
46             $error_message = $e->getMessage();
47             if (self::DB_DEBUG) echo "<p>An error occurred while connecting to the
48             database: $error_message </p>";
49         }
```

Step three for your class making the actual conection

- ☐ Add a public function to execute a select SQL Statement after the constructor.

```

50     public function select($query, $values = '') {
51
52         $statement = $this->pdo->prepare($query);
53
54         if (is_array($values)) {
55             $statement->execute($values);
56         } else {
57             $statement->execute();
58         }
59
60         $recordSet = $statement->fetchAll(PDO::FETCH_ASSOC);
61
62         $statement->closeCursor();
63
64         return $recordSet;
65     }
66
67 } // ends the class
68 ?>

```

Step four for your class adding a function to select records

- ☐ In top.php use the require_once statement to include the Database.php file using the LIB_PATH constant.

```

18
19 print '<!-- make Database connections -->';
20 require_once(LIB_PATH . '/Database.php');

```

Include your class file.

- ☐ In top.php create a database object connecting to your reader account

```

21
22 $thisDatabaseReader = new Database('rerickso_reader', 'r','RERICKSO_cs148_lab2');
23
24 ?>
25 </head>

```

Create the database object.

- ☐ sFTP your files to verify your code is not broken. Your pages should look the same. This just tests that you have connected to the database.
- ☐ Make a commit for Database.php

Display the database information.

Create your query.

- ☐ Create a file named sql.php with this structure: [sql.php](#) NOTE: you will replace my sql statement with yours.
- ☐ Create a query to select all the records from tblWildlife sorted by Common Name. Save this query in sql.php.

```
<?php
include 'top.php';
?>
<main>
    <p>Create Table SQL</p>

    <pre>
CREATE TABLE tblIssues(
    pmkIssuesId INT AUTO_INCREMENT PRIMARY KEY,
    fldActivity VARCHAR(40),
    fldEffects VARCHAR(200),
    fldImpact VARCHAR(200)
)
    </pre>

</main>
<?php include 'footer.php'; ?>
</body>
</html>
```

sample sql.php (note you will have select query not a create query)

- ☐ Confirm this query works by pasting it into phpMyAdmin and taking a screen shot of the records being sure to show the query in the screen shot (it shows above the records) saving the image as wildlife-query.png
- ☐ Make a commit for sql.php

index.php

- ☐ Create this block of code in index.php to retrieve the data to display.


```

index.php
1  <?php
2  include 'top.php';
3  $sql = 'SELECT pmkWildlifeId, fldType, fldCommonName, fldDescription, fldHabitat, ';
4  $sql .= 'fldReproduction, fldDiet, fldManagement, fldStatus, fldMainImage ';
5  $sql .= 'FROM tblWildlife ';
6  $sql .= 'ORDER BY fldCommonName';
7
8  $data = '';
9  $animals = $thisDatabaseReader->select($sql, $data);
10
11  ?>
12
13  <main>
14  <h2>Vermont's Wildlife</h2>

```

code to retrieve the data

- ☐ Use a foreach loop to display the image in a figure element.
- ☐ Set the image src to be the image you downloaded from the website
- ☐ Set the image alt to be the common name of the animal
- ☐ Set the figcaption to be the common name of the animal
- ☐ Make a commit for index.php

Style your page.

- ☐ Edit your css file as needed to have your index page display appropriately.
- ☐ Make a commit for your css
- ☐ Quality of your css. Are there major issues?
- ☐ Make sure you have pulled to the live site.
- ☐ Make sure you have sFTP pass.php to the live site.

Standard Requirements

- ☐ Main goals of assignment met.
- ☐ Be sure DEBUG is set to false.
- ☐ Update your main index.php file with links to all the files in the correct place (public and supporting).
- ☐ Do not use any banned elements (div, span, br).
- ☐ Be sure your text is still legible with and without the Color blindness tool.
- ☐ All files must use proper coding guidelines.

- ☐ Make sure paragraph, lists, forms text is not centered. Centering boxes is ok.
- ☐ (5% of your grade) Proper use of version control (commits on a regular basis, commits small with good message about what was accomplished with this commit) jigsaw.w3.org/css-validator/
- ☐ (5% of your grade) Make sure your files all pass w3c HTML validation: validator.w3.org/
- ☐ (5% of your grade) Make sure your files all pass w3c CSS validation: jigsaw.w3.org/css-validator/
- ☐ sFTP your files to your silk account which is our web server
- ☐ Submit your assignment only once in blackboard. See next direction first.
- ☐ Copy this code as a text submission in blackboard, look for the Write Submission button. You will need to click on the html button once you are in the editor and paste this code there:

```
<a target="_blank"
href="//lcarrera.w3.uvm.edu/cs148/live-lab2">
http://lcarrera.w3.uvm.edu/cs148/live-lab2</a>
```

- ☐ Update your main index to have a link to ADMIN/table-viewer.php?getDatabase= just after the link to your admin page.
- ☐ Create a file named credentials.php in your ADMIN folder to give us access to view your tables so we can grade them.

It should look like this but with your reader password:

```
<?php
$password = 'your reader password';
?>
```