Introduction:

This lab demonstrates how to collect data from an end-user via a web form and then save the data on a web server. Once the data are stored, we'll see how to read the data back. Storing and retrieving data are certainly primary functions in libraries, archives, museums, offices - everywhere - it is useful to complete the concept of the client/server architecture through practice reading the steps and actually doing it.

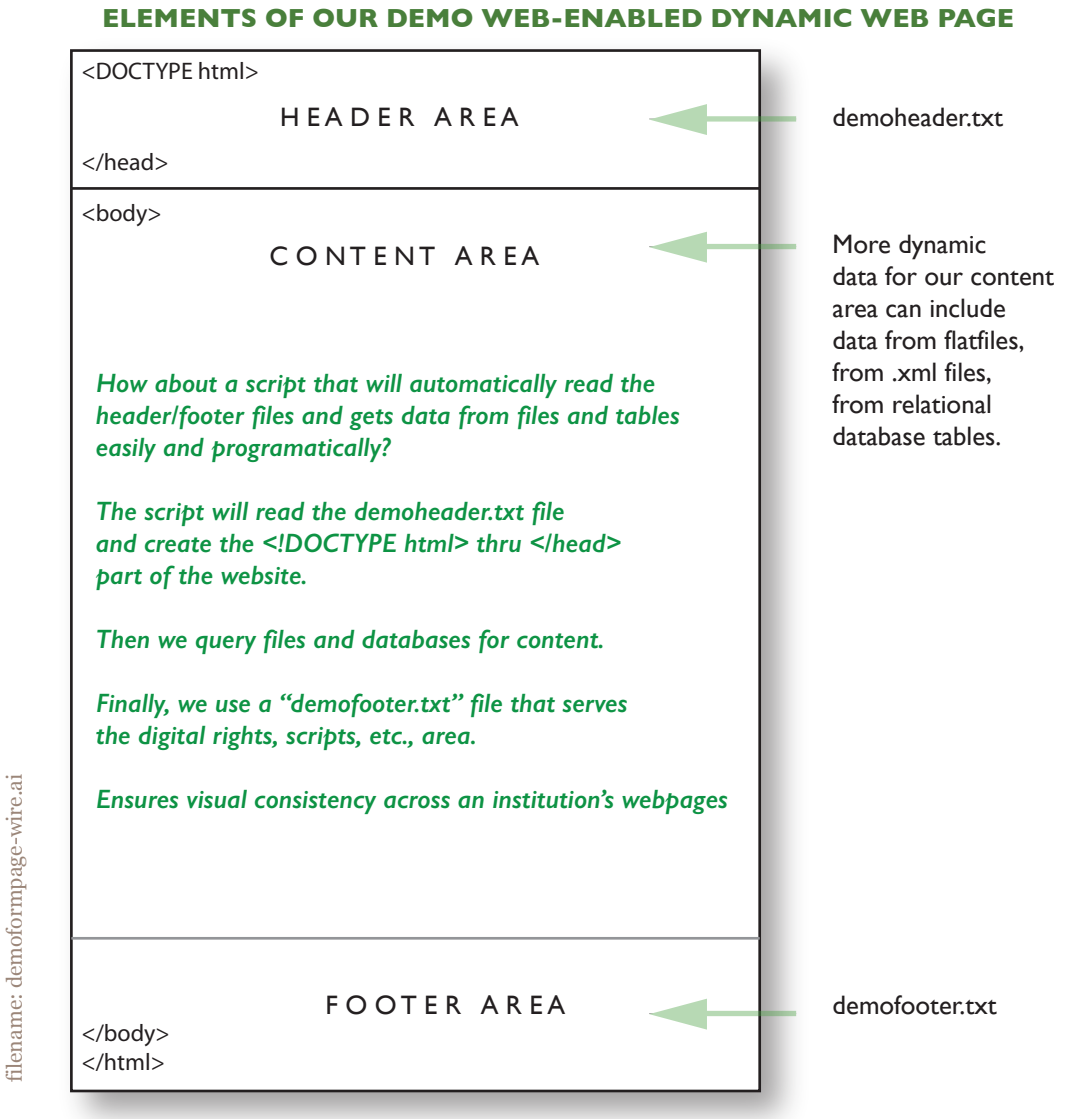
First: All files are available for you in an archive file (.zip). Here is the LINK. Download the folder, unzip the folder and then proceed. [http://web.simmons.edu/~benoit/unified/f16/dbdemo/studentfiles.zip]

Before we begin: what's needed - all in the .zip file:

①. A web form [to collect input from the end-user] in a webpage
②. A script on the server to collect the data, communicate with the web server itself [we'll use php as an example]
   1. A "demoheader.txt" file and a "demofooter.txt" file.
③. A flatfile on the server to save the data
④. A configuration file that holds our database access data config.ini
⑤. A relational database and table to store the data
⑥. Commands in the scripting language and SQL (the language of databases) to select data and send back to the end-user.

The purpose of this lab is to demonstrate the integration of most of our tech skills. We'll read text files, communicate with relational databases and other resources … This lab emulates real practices.

Overview:



How we communicate with the server: Recall from the first week's study of the Client/Server Architecture. This lab completes the image of accepting data from patrons, communicating with server-based resources, and returning the data to the end-user.

For more, take the Relational Database class (458) and Web Development/Information Architecture (467) classes.

How the process works:

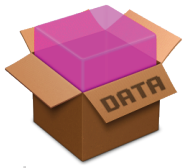
1 Fill in the form


```
<form name="myform"
    action="myscript.php"
    method="post">
<input type="username">
```

2 Transmit data by pressing submit button

submit

3 Data are "packaged" and sent by "post" or "get" to the web server



4 Data are received by the web server and the package is open to extract the data...

E.g., webserver is running Apache software

Database Server (MySQL)

5 Script/Program named in the "action" part of form runs `myscript.php`

E.g., myscript.php is on the web server ... starts when web server receives the "action" statement

6 Data from form is copied into the script or program, e.g.,

```
<?php
$name = $_POST["username"];
```

create variable in the PHP script

data extracted from the web form

7 The script on the server executes the commands ... Example with php


```
$dataToSave = $patronName . " " . $patronEmail;
echo file_put_contents("demofile.txt", $dataToSave, FILE_APPEND);
```

8 Store the data in a rdbms - concatenate SQL commands with variables to create an actual command:


```
$sql = "INSERT INTO myTable FIELDS (name,email)
     VALUES ('" . $patronName . "','". $patronEmail . "')");
```

9 Let's read the text file and send the data back to the patron's web browser


```
echo file_get_contents("demofile.txt");
```

10 Same with the DB: select the data and send back to the end-user's browser


```
$sql = "SELECT * FROM myTable WHERE name = '" .$patronName . "'";
```

Data from the SQL statement are stored in a "result" and the data are then extracted from the result, packaged for the Net, and returned to browser.

filename: demoformdb-Nov25-2.ai

1. Creating the demoprogramfile.txt and config.ini

Using your text editor

①. Create an empty file called `demoprogramfile.txt` and then upload it to your webspace (e.g., ~smithj/lis488/demoprogramfile.txt).

②. Edit the downloaded config.ini file - copy it *exactly* - no extra spaces, no extra lines, no typos… Change the placeholders (like "YOURSIMMONSUSERNAME") with your actual username. Password is your student id number. Database name will be provided in class. Save the file as `config.ini` file and upload it to your webspace.

```
[database]
username = YOURSIMMONSUSERNAME
password = YOURSTUDENTID
dbname = YOURDATABASE
```

Example:

```
[database]
username = benoit
password = 3939989
dbname = lis48801_benoit
```

Upload the demoform.php script to your webserver space.

①. First click on the link to download the .php file to your computer's hard disk and then …

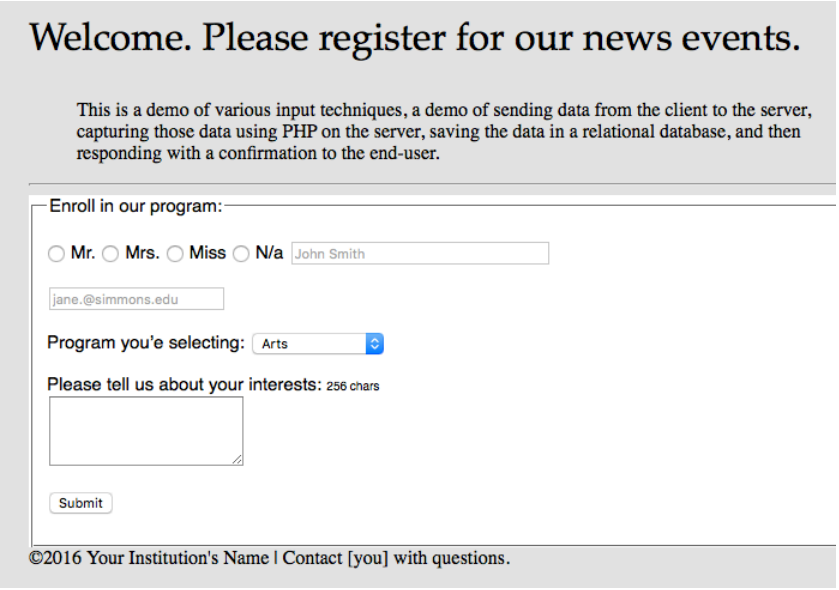
②. Use your SFTP software, upload the file to your web space on the server

2. Create the Web form

Having prepared for data on the server side, let's make the client-side webform for input.

You can copy this example - but read the source, too, to understand the relationship of the name of the input field and how it relates to the name-value pair being sent from the user's browser to the script on the server.

Here's a link to the source: http://web.simmons.edu/~benoit/unified/f16/dbdemo/demoform.html. Check out the .html form you downloaded in the .zip folder. Upload your demoform.html to your webspace.



Create your database table:

Finally, we're ready to communicate with the webserver and from there to the database server. You'll need your terminal window (Unix on the Mac or use Putty or WinSTP to connect to web.simmons.edu).

Locate a terminal window (on the Mac) by navigating to the "Utilities" folder.

Read carefully: You'll need your Simmons ID and email username to complete this.

① Open a terminal window (on the Macs in the lab).
② This document describes how to connect to MySQL using a terminal window.

① You need your email name, email password, and your Simmons ID number.
② Once you start the terminal window, you'll see the prompt $
③ The commands for you to type are highlighted but be sure to read every line very carefully and note the spaces, the capitalization, the use of semi-colon to end the SQL commands.
④ You'll receive a variety of responses on the screen that'll vary from the demo.
⑤ The instructions are in this italic font; the commands for you to type are in the monospace on yellow.
⑥ The first part will establish the connection with the webserver. (-l is the letter "l", not the number 1)
⑦ After the first line, you'll enter your password - this is your email password. The password will *not* show up on the screen. **REPLACE** `benoit` **WITH YOUR USERNAME!**

```
$ ssh -l benoit web.simmons.edu
benoit@web.simmons.edu's password:
```

```
Last login: Fri Aug 19 10:30:43 2016 from c-73-149-17-22.hsd1.ma.comcast.net
```

When connected you want to go from the webserver to the MySQL database server.
You'll need your Simmons ID as the SQL password in response to the Enter password: prompt.

```
[benoit@cleo ~]$ mysql -u benoit -h dany.simmons.edu -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 24657
Server version: 5.1.73 Source distribution

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

Type this command to see what databases you have access to.

```
MySQL [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| gb_demo            |
+--------------------+
5 rows in set (0.15 sec)
```

If you do not have a database, let me know right away. Only the system administrator can create databases for us in this class. If you have a database, the name may be something like "smith_488". For demonstration, let's say it is. Only one database at a time can be active; select the database with the USE command. If you make a mistake typing the command end it with the semi-colon and start again.

```
MySQL [(none)]> use smith_488;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
```

It's useful to see if you have any tables already. Probably you don't; so let's make one.

```
MySQL [gb_demo]> show tables;
+----------------------+
| Tables_in_gb_demo    |
+----------------------+
| answerTable          |
+----------------------+
1 row in set (0.00 sec)
```

To create our table we issue the CREATE TABLE command. The last line of the SQL command must have a

semi-colon. You can enter your command on multiple lines - press the return key after each line.
Pay close attention to semi-colons, spaces, and capitalization! This is case-sensitive: capitalization and spaces are vital. Notice that after each line you press the enter key. Until you press the semi-colon, SQL will keep asking you for more, indicated by the ->

```
MySQL [gb_demo]> CREATE TABLE demotable (
    -> recno INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    -> salut VARCHAR(4),
    -> patronName VARCHAR(30) NOT NULL,
    -> patronEmail VARCHAR(30),
    -> programChoice VARCHAR(20),
    -> patronComments VARCHAR(256),
    -> createdOn TIMESTAMP
    -> );
Query OK, 0 rows affected (0.10 sec)

MySQL [gb_demo]>
```

When you're done, we exit both the SQL server and then the webserver.

```
MySQL [gb_demo]> exit
Bye
[benoit@cleo ~]$ exit
logout
```

All Done

Test your webpage! If your username is "smithj" then your form should be available at http://web.simmons.edu/~smithj/lis488/demoform.html

Try it and see what works. If it doesn't work, try another browser; if it still doesn't work - let's debug!

Appendix: hard-copy versions of the config.ini and .php files.

```
<?php
/* make sure your can access your database and have created the
table to capture these data before doing this exercise.

The demoform.html web form calls this script.
*/

/* ****************************************
      GET THE VARIABLES FROM THE END-USER
      - in this example, I used the tab to line up the variables.
**************************************** */
$salutation = $_POST["salutation"];
$name       = $_POST["username"];
$email      = $_POST["email"];
$program    = $_POST["program"];
$comments   = $_POST["comments"];

/* ****************************************
Before continuing, let's send data back to the end-user
to create an html page in the user's browser.  All the data
and error messages are going to be sent back to the browser
 so we should make it value html.
**************************************** */
/* echo "<!DOCTYPE html><html><head><title>Program Confirmation</title>"
       "</head>\n<body>";
*/
/* NOTICE WE CAN HARD CODE THE HEADER (above) OR USE A FILE (below) */
echo file_get_contents("demoheader.txt");

echo "DEBUGGING LINES ARE INCLUDED - REMOVE AS YOU WISH<br />";
echo "Variables from the form<ol>
      <li>$salutation</li>
      <li>$name</li>
      <li>$email</li>
      <li>$program</li>
      <li>$comments</li></ol>";

/* let's store the data in a flat file using PHP */

/* ****************************************
first create a variable ($dataToSave) to hold all the data.
the next line concatenates the variables from the end-user's form
and creates one giant string of data ... that string is stored
in a single variable, called $dataTosave

Notice we can mix-and-match: spaces, html tags, and data from the form
**************************************** */
$dataToSave = $salutation . " " . $name . " " . $email . " " . $program;
$dataToSave = $dataToSave . "\n<p>$comments</p><hr />";

echo "<p>For debugging only: $dataToSave</p>";

/* ****************************************
we save the data to a file - appending the data
```

```
****************************************** */
file_put_contents("demoprogramfile.txt", $s, FILE_APPEND);

/* ****************************************
Now let's insert the data in a relational database table
// http://www.w3schools.com/PHP/php_mysql_connect.asp
****************************************** */

/* ****************************************
Need to tell SQL where to save the data
****************************************** */
/* THE DATA ARE NOW STORED IN A config.ini FILE -
USING AN .INI FILE MAKES IT EASY TO DEVELOP AND THEN
DISTRIBUTE PROGRAMS */

$config = parse_ini_file("config.ini");
// notice the "parse_ini_file" parses the config file and stores the values in
// an array.  Then we use the array for database access.

/* ****************************************
Make a connection between the web server and the database server
****************************************** */
$conn = new mysqli('dany.simmons.edu', $config['username'], $config['password'],
$config['dbname']);

/* ****************************************
Check that we have our connection; if so, continue,
else stop running the program (that's the "die()" command)
****************************************** */

if ($conn->connect_error) {
      die("Sorry, the connection could not be made: " . $conn->connect_error);
}
echo "Okay, we are connected to the db.<br />";


/* ****************************************
If we're okay to proceed, let's create the command that SQL can
understand by integrating the SQL parts and the data from the
end-user into the INSERT statement
****************************************** */

// Save the data in your table (demotable)
// First prepare an SQL command
$sql = "INSERT INTO demotable (salut, patronName, patronEmail, programChoice,
patronComments) VALUES ($salutation, $name, $email, $program, $comments)";

// Now let's use that command
/* ****************************************
Now let's use that command.  If the record is inserted over
the connection ($conn->query($sql)) and a new record is
created, SQL returns a boolean value TRUE.
If true, we tell the user new record, else an error
****************************************** */

if ($conn->query($sql) === TRUE) {
      echo  "New Record for " . $dataToSave;
} else {
```

```
      echo "Sorry, there was an error " .$sql . $conn->error;
}

/* ****************************************
Let's test our database, too, by issuing a SELECT statement
to pull the data back out of the database table.
We'll recycle the $sql variable ...
Pay -very- close attention to the single and double quotes.
**************************************** */
$sql = "SELECT * FROM demotable WHERE patronName = '" . $name . "'";

/* ****************************************
We issue the command ($sql) and store the results from the table
in a variable, called $result
**************************************** */
$result = $conn->query($sql);

if ($result->num_rows > 0) {
      // if we have > 0 rows, it means we have successful search
      while ($row = $result->fetch_assoc()) {
            echo "record number " . $row["recno"]. " - Name: " .
            $row["patronName"]. " " . $row["patronEmail"] . " " .
            $row["programChoice"] . " " . $row["patronComments"] . "<br />";
      }
} else {
      echo "Sorry, no records.";
}

// Close the connection to the DB
$conn->close();

/* ****************************************
We're all done with the database part.  Let's confirm the file
we wrote, too.
**************************************** */
echo "<p>Here we confirm the file we created above.</p>";

/* ****************************************
First let's make sure the file exists!
**************************************** */
if (file_exists("demoprogramfile.txt")) {
      echo file_get_contents("demoprogramfile.txt");
} else {
      echo "Sorry, the demoprogramfile does not exist or access permission denied.";
}

/* ****************************************
We're all done!  let's close out the webpage.
**************************************** */
// echo "</body></html>";

echo file_get_contents("demofooter.txt");

/* **** and close the php file **** */
?>
```