INFO 151
Web Systems and Services

Week 7 (Lab)

Dr Philip Moore

Dr Zhili Zhao

PHP and Database Exercises

The Course Textbook

- Chapters 17 and 18 cover MySQL and the related PHP scripting
- These chapters cover the work introduced in this course
- At the end of Chapters 17 and 18 there are:
  - Questions and Answers (Q & A)
  - Workshop (Questions with answers)
  - Exercises
- Complete the Q & A, workshop, and exercises

Runtime Errors

Catching Runtime Errors

- When executing JavaScript or PHP program code different runtime errors can occur - errors include
  - Fail to connect to the database
  - Fail to run the SQL statement
  - Invalid input by a user (including criminal attempts to access a database)
  - Runtime errors (network issues of server down)
- To catch errors use:
  - The **`try`** statement tests a block of code for runtime errors
  - The **`catch`** statement handles the runtime error
  - The **`throw`** statement creates custom errors
  - The **`finally`** statement execute JavaScript program code after the try and catch methods regardless of the result

Try…catch block

- The **`try`** statement defines a block of code to be tested for errors while it is being executed (at ***runtime***)
- The **`catch`** statement defines a block of code to be executed if an error occurs in the **`try`** block.
- The statements **`try`** and **`catch`** come in pairs

```
try {
    Block of code to try
}
catch(err) {
    Block of code to handle errors
}
```

try…catch…finally block

- The syntax for a **try…catch..finally** block

```
try {
      statements
 }
catch ( arguments ) {
}
finally {
      statements
}
```

Source History

```
9          <meta name="viewport" content="width=device-width, initial-scale=1.0">
10      </head>
11      <body>
12          <p>Please input a number between 5 and 10:</p>
13          <input id="demo" type="text">
14          <button type="button" onclick="myFunction()">Test Input</button>
15          <p id="p01"></p>
16          <script>
17              function myFunction() {
18                  var message, x;
19                  message = document.getElementById("p01");
20                  message.innerHTML = "";
21                  x = document.getElementById("demo").value;
22                  try {
                        if(x == "")  throw "is empty";
24                      if(isNaN(x)) throw "is not a number";
25                      x = Number(x);
26                      if(x > 10)   throw "is too high";
27                      if(x < 5)    throw "is too low";
28                  }
29                  catch(err) {
30                      message.innerHTML = "Input " + err;
31                  }
32                  finally {
33                      document.getElementById("demo").value = "";
34                  }
35              }
36          </script>
37      </body>
38  </html>
```

The value is set to an empty string

PHP Exercises

Exercises

- Write an HTML program using both JavaScript and PHP **`for`** loops
- The output should be a 12 times table with the output as follows:
  - A times table (a heading)
  - 1 x 1 = 1 (the program output)
  - 1 x 2 = 2
  - 1 x 3 = 3 (etc up to 1 x 12 = 12))
- Repeat the exercise using JavaScript and PHP **`while`** loops
- Repeat the exercise using JavaScript and PHP **`do`**…**`while`** loops
- Extend the programs to output all the times tables from 1 to 12

Source History

```php
<?php
$dbhost = 'localhost:3306'; $dbuser = 'philip';
$dbpass = 'philip'; $db = 'test';
//SQL statements: $sql and $sqla
$sql = 'SELECT id, f_name, s_name, email FROM contacts';
$sqla = 'SELECT * FROM contacts';
//connect MySQL database
$conn = mysqli_connect($dbhost, $dbuser, $dbpass);
if(!$conn) {
    Die_error('Connection failed!: '.mysqli_errno());
}
echo 'Database connected <br>';
mysqli_select_db($conn, $db);
echo 'database selected: ' .$db.'<br>';
echo ($sqla.'<br>'.'-------------------<br>');
$retval = mysqli_query($conn, $sqla);
if(!$retval) {
    die('Could not get data');
}
while($row = mysqli_fetch_assoc($retval)) {
    echo "id: {$row['id']}<br>".
         "f_name: {$row['f_name']}<br>".
         "s_name: {$row['s_name']}<br>".
         "email:  {$row['email']}<br>".
         "-----------------------<br>";
}
mysqli_free_result($retval);
echo "fetched data successfully\n";
mysqli_close($conn);
?>
```

The test database used in the example PHP script

SQL statements $sql / $sqla

SQL statement used $sqla

Lab Exercises

- The PHP script as used and shown in the previous slide:
  - Is a simple solution to implement a connection to a MySQL (or other relational database systems) and run SQL queries on the database
- In practice the script would be improved using:
  - **`try`**…**`catch`**…**`finally`** block
  - Improve the PHP script shown by adding **`try`**…**`catch`**…**`finally`** blocks to:
    - The database connection: if(!$conn) {…} (with the error message)
    - The data retrieval : if(!retrieval) {…} (with the error message)

SQL Exercises

Lab Exercises

- Create your personal account in the MySQL Administration Tool:
  - Username
  - Password
  - Global privileges (in this case you will be the SysAdmin
- Create a database in NetBeans
  - Call it “test”
- Create a ”Contacts” table with four attributes (columns) and four records (rows)
- Populate the table with data values
- Run your database with SQL queries using the PHP script

Exercises

- Implement the following SQL statements
- The following SQL statements to be written into your PHP script and run on the test database:
  - INSERT
  - ADD
  - SELECT (all values and selected values)
  - UPDATE records
  - REPLACE INTO table_name (column list) VALUES (column values);
  - DELETE FROM table_name [WHERE some_condition_is_true] [LIMIT rows]

Exercises

- Complete the practical exercises in the *PHP Practical Exercises* document