PHP MySQL: Querying Data from Database

Summary: in this lab, you will learn how to query data from MySQL database by using PHP PDO. You will also learn how to use PDO prepared statement to select data securely.

To query data from the MySQL database, follow the steps below:

First, connect to a MySQL database. (As you did on Lab01)

```php
$pdo = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);
```

Then, construct a SELECT statement and execute it by using the query() method of the PDO object.

```php
$sql = 'SELECT lastname, firstname, jobtitle
        FROM employees
        ORDER BY lastname';

$q = $pdo->query($sql);
```

Next, set the PDO::FETCH_ASSOC fetch mode for the PDOStatement object by using the setFetchMode()method. The PDO::FETCH_ASSOC mode instructs the fetch() method to return a result set as an array indexed by column name.

```php
$q->setFetchMode(PDO::FETCH_ASSOC);
```

After that, fetch each row from the result set until there is no row left by using the fetch() method of the PDOStatement object.

```
<table class="table table-bordered table-condensed">
    <thead>
        <tr>
            <th>First Name</th>
            <th>Last Name</th>
            <th>Job Title</th>
        </tr>
    </thead>
    <tbody>
        <?php while ($r = $q->fetch()): ?>
            <tr>
                <td><?php echo htmlspecialchars($r['lastname']) ?></td>
                <td><?php echo htmlspecialchars($r['firstname']); ?></td>
                <td><?php echo htmlspecialchars($r['jobtitle']); ?></td>
            </tr>
        <?php endwhile; ?>
    </tbody>
</table>
```

Putting it all together.

Note 1: Change line 23 to: <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css">

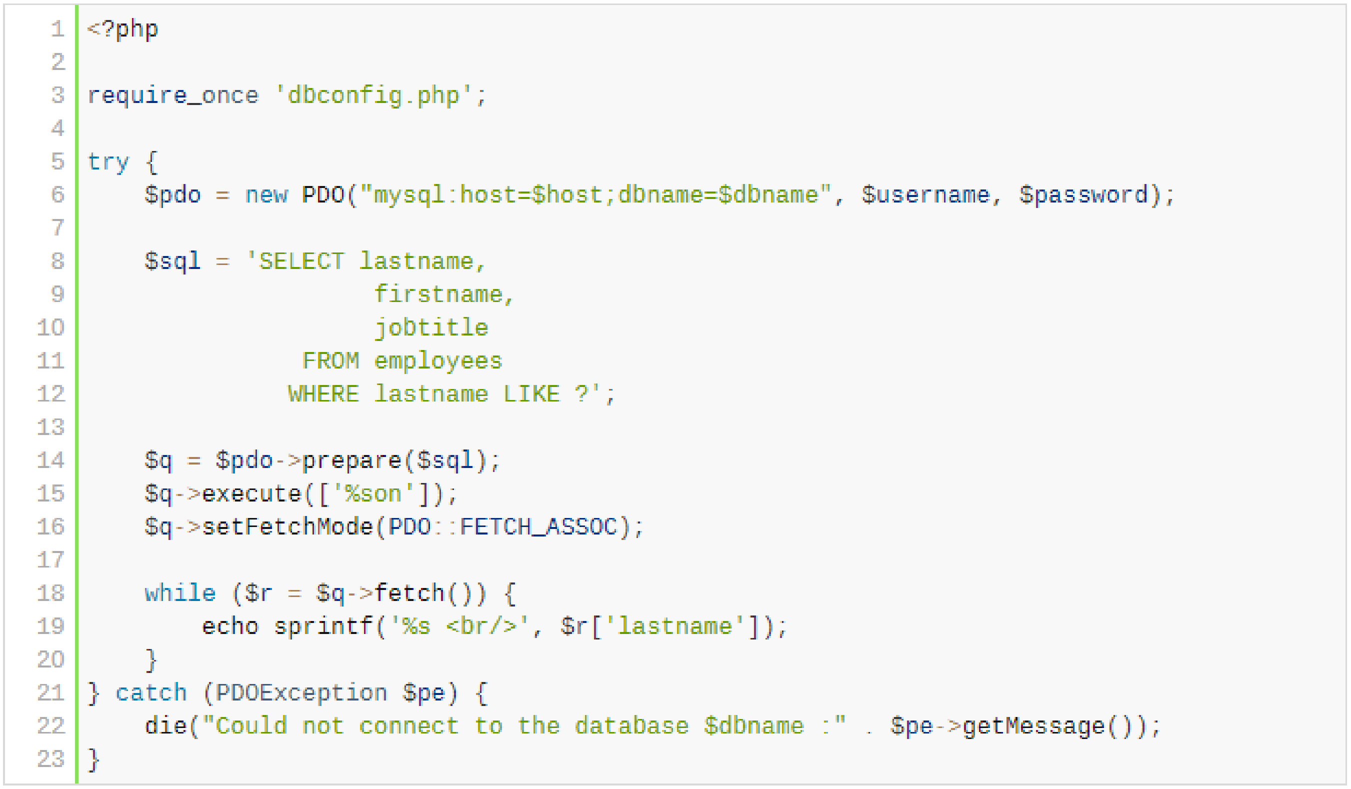
Note 2: Delete line 24

```php
<?php
require_once 'dbconfig.php';

try {
    $pdo = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);

    $sql = 'SELECT lastname,
                    firstname,
                    jobtitle
               FROM employees
              ORDER BY lastname';

    $q = $pdo->query($sql);
    $q->setFetchMode(PDO::FETCH_ASSOC);
} catch (PDOException $e) {
    die("Could not connect to the database $dbname :" . $e->getMessage());
}
?>
<!DOCTYPE html>
<html>
    <head>
        <title>PHP MySQL Query Data Demo</title>
        <link href="css/bootstrap.min.css" rel="stylesheet">
        <link href="css/style.css" rel="stylesheet">
    </head>
    <body>
        <div id="container">
            <h1>Employees</h1>
            <table class="table table-bordered table-condensed">
                <thead>
                    <tr>
                        <th>First Name</th>
                        <th>Last Name</th>
                        <th>Job Title</th>
                    </tr>
                </thead>
                <tbody>
                    <?php while ($row = $q->fetch()): ?>
                        <tr>
                            <td><?php echo htmlspecialchars($row['lastname']) ?></td>
                            <td><?php echo htmlspecialchars($row['firstname']); ?></td>
                            <td><?php echo htmlspecialchars($row['jobtitle']); ?></td>
                        </tr>
                    <?php endwhile; ?>
                </tbody>
            </table>
    </body>
</div>
</html>
```

Part 2: PHP MySQL Querying data using PDO prepared statement

In practice, we often pass the argument from PHP to the SQL statement e.g., get the employee whose last name ends with son . To do it securely and avoid SQL injection attack, you need to use the PDO prepared statement.

Let’s take a look at the following example:



```php
<?php

require_once 'dbconfig.php';

try {
    $pdo = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);

    $sql = 'SELECT lastname,
                    firstname,
                    jobtitle
               FROM employees
              WHERE lastname LIKE ?';

    $q = $pdo->prepare($sql);
    $q->execute(['%son']);
    $q->setFetchMode(PDO::FETCH_ASSOC);

    while ($r = $q->fetch()) {
        echo sprintf('%s <br/>', $r['lastname']);
    }
} catch (PDOException $pe) {
    die("Could not connect to the database $dbname :" . $pe->getMessage());
}
```

(Check the sprintf() Function at https://www.w3schools.com/php/func_string_sprintf.asp)

How the script works:

- First, we use a question mark (?) in the SELECT statement. PDO will replace the question mark in the query by the corresponding argument. The question mark is called positional placeholder.
- Next, we call the prepare() method of the PDO object to prepare the SQL statement for the execution.
- Then, we execute the statement by calling the execute() method of the PDOStatement object. In addition, we pass an argument as an array to replace the placeholder in the SELECT statement. By doing this, the SELECT statement will be translated as follows:

```sql
SELECT lastname,
       firstname,
       jobtitle
  FROM employees
 WHERE lastname LIKE '%son';
```

- After that, we set the fetch mode for the PDOStatement object.
- Finally, we fetch each row of the result set and display the last name field.

PHP provides you with another way to use placeholders in the prepared statement called named placeholder. The advantages of using the named placeholder are:

- More descriptive.
- If the SQL statement has multiple placeholders, it is easier pass the arguments to the execute()method.

Let’s take a look at the following example:

```php
<?php

require_once 'dbconfig.php';

try {
    $pdo = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);

    $sql = 'SELECT lastname,
                    firstname,
                    jobtitle
               FROM employees
              WHERE lastname LIKE :lname OR
                    firstname LIKE :fname;';

    // prepare statement for execution
    $q = $pdo->prepare($sql);

    // pass values to the query and execute it
    $q->execute([':fname' => 'Le%',
        ':lname' => '%son']);

    $q->setFetchMode(PDO::FETCH_ASSOC);

    // print out the result set
    while ($r = $q->fetch()) {
        echo sprintf('%s <br/>', $r['lastname']);
    }
} catch (PDOException $e) {
    die("Could not connect to the database $dbname :" . $e->getMessage());
}
```

The :lname and :fname are the named placeholders. They are substituted by the corresponding argument in the associative array that we pass to the execute method.

In this lab, you have learned how to query data from MySQL database using PDO objects.