Chapter 19

Professional PHP for working with MySQL

Objectives

Applied

1. Use PHP Data Objects (PDO) with prepared statements to develop new database-driven web applications.
2. Use the mysqli extension to develop new database-driven web applications.

Knowledge

1. Distinguish between the use of PHP Data Objects (PDO) and the use of the mysqli extension.
2. Describe the use of prepared statements when working with PDO, including the use of named and question mark parameters.
3. Describe the use of the exception error mode when working with PDO.
4. Describe the use of mysqli in both its procedural and object-oriented styles.

PDO (PHP Data Objects)

Pros

- Is included with PHP 5.1 and later and available for 5.0.
- Provides an object-oriented interface.
- Provides a consistent interface that's portable between other database servers such as Oracle, DB2, and Microsoft SQL Server.
- Takes advantage of most new features found in MySQL 4.1.3 and later.

Cons

- Doesn't work with versions of PHP 4.x, 3.x, or earlier.
- Doesn't take advantage of some advanced features found in MySQL 4.1.3 and later, such as multiple statements.

mysqli (MySQL improved extension)

Pros

- Is included with PHP 5 and later.
- Provides both an object-oriented interface and a procedural interface.
- Takes advantage of all new features found in MySQL 4.1.3 and later.

Cons

- Can’t be used with other database servers.

Recall from Chapter 4 dynamic SQL statements:

A query method with the SELECT statement in a variable

```
$query = 'SELECT * FROM products
          WHERE categoryID = 1
          ORDER BY productID';
$products = $db->query($query);
```

A query method with the SELECT statement as the argument

```
$products = $db->query('SELECT * FROM products');
```

In practice, use *prepared SQL statements* rather than dynamic SQL statements for improved performance and security

How to use the fetchAll method to return a result set

```
$query = 'SELECT * FROM products';
$statement = $db->prepare($query);
$statement->execute();
$products = $statement->fetchAll();
$statement->closeCursor();
foreach ($products as $product) {
    echo $product['productName'] . '<br />';
}
```

How to use the fetch method to loop through a result set

```
$query = 'SELECT * FROM products';
$statement = $db->prepare($query);
$statement->execute();
$product = $statement->fetch();       // get first row
while ($product != null) {
    echo $product['productName'] . '<br />';
    $product = $statement->fetch();   // get next row
}
$statement->closeCursor();
```

Some methods of the PDO class

```
prepare($sql_statement)
lastInsertId()
```

Some methods of the PDOStatement class

```
bindValue($param, $value)
execute()
fetchAll()
fetch()
rowCount()
closeCursor()
```

How to use named parameters

```
$category_id = 1;
$price = 400;

$query = 'SELECT * FROM products
          WHERE categoryID = :category_id
          AND listPrice > :price';
$statement = $db->prepare($query);
$statement->bindValue(':category_id', $category_id);
$statement->bindValue(':price', $price);
$statement->execute();
$products = $statement->fetchAll();
$statement->closeCursor();
```

How to use question mark (unnamed) parameters

```
$query = 'SELECT * FROM products
          WHERE categoryID = ?
          AND listPrice > ?';
$statement = $db->prepare($query);
$statement->bindValue(1, $category_id);
$statement->bindValue(2, $price);
$statement->execute();
$products = $statement->fetchAll();
$statement->closeCursor();
```

How to modify data

```
// Sample data
$category_id = 2;
$code = 'hofner';
$name = 'Hofner Icon';
$price = '499.99';

// Prepare and execute the statement
$query = 'INSERT INTO products
      (categoryID, productCode, productName, listPrice)
          VALUES
      (:category_id, :code, :name, :price)';
$statement = $db->prepare($query);
$statement->bindValue(':category_id', $category_id);
$statement->bindValue(':code', $code);
$statement->bindValue(':name', $name);
$statement->bindValue(':price', $price);
$success = $statement->execute();
$row_count = $statement->rowCount();
$statement->closeCursor();
```

How to modify data (continued)

```
// Get the last product ID that was generated
$product_id = $db->lastInsertId();

// Display a message to the user
if ($success) {
    echo "<p>$row_count row(s) was inserted
              with this ID: $product_id</p>";
} else {
    echo "<p>No rows were inserted.</p>";
}
```

The three error modes for PDO

```
ERRMODE_SILENT  // default mode for executing
ERRMODE_WARNING
ERRMODE_EXCEPTION  //default mode for connecting to db
```

Setting the error mode with the PDO constructor

```
$dsn = 'mysql:host=localhost;dbname=my_guitar_shop1';
$username = 'mgs_user';
$password = 'pa55word';
$options = array(PDO::ATTR_ERRMODE =>
PDO::ERRMODE_EXCEPTION);

try {
    $db = new PDO($dsn, $username, $password, $options);
} catch (PDOException $e) {
    $error_message = $e->getMessage();
    echo "<p>Error connecting to database:
            $error_message </p>";
    exit();
}
```

Setting the mode with the setAttribute method

```
$db->setAttribute(
        PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

How to catch PDOException objects

```
try {
    $query = 'SELECT * FROM product';
    $statement = $db->prepare($query);
    $statement->execute();
    $products = $statement->fetchAll();
    $statement->closeCursor();
} catch (PDOException $e) {
    $error_message = $e->getMessage();
    echo "<p>Database error: $error_message </p>";
    exit();
}
```

The model/database.php file

```
<?php
$dsn = 'mysql:host=localhost;dbname=my_guitar_shop2';
$username = 'mgs_user';
$password = 'pa55word';
$options =
    array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION);

try {
    $db = new PDO($dsn, $username, $password, $options);
} catch (PDOException $e) {
    $error_message = $e->getMessage();
    include 'errors/db_error_connect.php';
    exit;
}

function display_db_error($error_message) {
    global $app_path;
    include 'errors/db_error.php';
    exit;
}
?>
```

The model/category_db.php file

```
<?php
function get_categories() {
    global $db;
    $query = 'SELECT * FROM categories
              ORDER BY categoryID';
    try {
        $statement = $db->prepare($query);
        $statement->execute();
        $result = $statement->fetchAll();
        $statement->closeCursor();
        return $result;
    } catch (PDOException $e) {
        display_db_error($e->getMessage());
    }
}
```

The model/category_db.php file (continued)

```
function get_category($category_id) {
    global $db;
    $query = 'SELECT * FROM categories
              WHERE categoryID = :category_id';
    try {
        $statement = $db->prepare($query);
        $statement->bindValue(':category_id', $category_id);
        $statement->execute();
        $result = $statement->fetch();
        $statement->closeCursor();
        return $result;
    } catch (PDOException $e) {
        display_db_error($e->getMessage());
    }
}
?>
```

The model/product_db.php file

```
<?php
function get_products_by_category($category_id) {
    global $db;
    $query = 'SELECT * FROM products
              WHERE categoryID = :category_id
              ORDER BY productID';
    try {
        $statement = $db->prepare($query);
        $statement->bindValue(':category_id', $category_id);
        $statement->execute();
        $result = $statement->fetchAll();
        $statement->closeCursor();
        return $result;
    } catch (PDOException $e) {
        $error_message = $e->getMessage();
        display_db_error($error_message);
    }
}
```

The model/product_db.php file (continued)

```
function get_product($product_id) {
    global $db;
    $query = 'SELECT *
              FROM products
              WHERE productID = :product_id';
    try {
        $statement = $db->prepare($query);
        $statement->bindValue(':product_id', $product_id);
        $statement->execute();
        $result = $statement->fetch();
        $statement->closeCursor();
        return $result;
    } catch (PDOException $e) {
        $error_message = $e->getMessage();
        display_db_error($error_message);
    }
}
```

The model/product_db.php file (continued)

```
function add_product($category_id, $code, $name,
        $description, $price, $discount_percent) {

    global $db;
    $query = 'INSERT INTO products
        (categoryID, productCode, productName, description,
                  listPrice, discountPercent, dateAdded)
              VALUES
        (:category_id, :code, :name, :description, :price,
                  :discount_percent, NOW())';
```

The model/product_db.php file (continued)

```
    try {
        $statement = $db->prepare($query);
        $statement->bindValue(':category_id', $category_id);
        $statement->bindValue(':code', $code);
        $statement->bindValue(':name', $name);
        $statement->bindValue(':description', $description);
        $statement->bindValue(':price', $price);
        $statement->bindValue(':discount_percent',
                              $discount_percent);
        $statement->execute();
        $statement->closeCursor();

        // Get the last product ID that was generated
        $product_id = $db->lastInsertId();
        return $product_id;
    } catch (PDOException $e) {
        $error_message = $e->getMessage();
        display_db_error($error_message);
    }
}
```

The model/product_db.php file (continued)

```
function update_product($product_id, $code, $name,
    $description,$price, $discount_percent, $category_id) {

    global $db;
    $query = 'UPDATE Products
              SET productName = :name,
                  productCode = :code,
                  description = :description,
                  listPrice = :price,
                  discountPercent = :discount_percent,
                  categoryID = :category_id
              WHERE productID = :product_id';
```

The model/product_db.php file (continued)

```
    try {
        $statement = $db->prepare($query);
        $statement->bindValue(':name', $name);
        $statement->bindValue(':code', $code);
        $statement->bindValue(':description', $description);
        $statement->bindValue(':price', $price);
        $statement->bindValue(':discount_percent',
                              $discount_percent);
        $statement->bindValue(':category_id', $category_id);
        $statement->bindValue(':product_id', $product_id);
        $row_count = $statement->execute();
        $statement->closeCursor();
        return $row_count;
    } catch (PDOException $e) {
        $error_message = $e->getMessage();
        display_db_error($error_message);
    }
}
```

The model/product_db.php file (continued)

```
function delete_product($product_id) {

    global $db;
    $query = 'DELETE FROM products
              WHERE productID = :product_id';
    try {
        $statement = $db->prepare($query);
        $statement->bindValue(':product_id', $product_id);
        $row_count = $statement->execute();
        $statement->closeCursor();
        return $row_count;
    } catch (PDOException $e) {
        $error_message = $e->getMessage();
        display_db_error($error_message);
    }
}
?>
```

How to use mysqli to connect to a MySQL database (object-oriented)

```
$host = 'localhost';
$username = 'mgs_user';
$password = 'pa55word';
$db_name = 'my_guitar_shop1';
@ $db = new mysqli(
    $host, $username, $password, $db_name);
```

How to use mysqli to connect to a MySQL database (procedural)

```
$host = 'localhost';
$username = 'mgs_user';
$password = 'pa55word';
$db_name = 'my_guitar_shop1';
@ $db = mysqli_connect(
    $host, $username, $password, $db_name);
```

Two properties of the mysqli object for checking connection errors

```
connect_errno
connect_error
```

How to check for a connection error (OO)

```
$connection_error = $db->connect_error;
if ($connection_error != null) {
    echo "<p>Error connecting to database:
          $connection_error</p>";
    exit();
}
```

How to check for a connection error (procedural)

```
$connection_error = mysqli_connect_error();
if ($connection_error != null) {
    echo "<p>Error connecting to database:
          $connection_error</p>";
    exit();
}
```

A mysqli method for returning a result set

```
query($select_statement)
```

A property and a method of the mysqli_resultset class

```
num_rows
fetch_assoc()
```

How to execute a SELECT statement with mysqli

```
// Execute the statement
$category_id = 2;
$query = "SELECT * FROM products
          WHERE categoryID = $category_id";
$result = $db->query($query);

// Check the result set
if ($result == false) {
    $error_message = $db->error;
    echo "<p>An error occurred: $error_message</p>";
    exit();
}

// Get the number of rows in the result set
$row_count = $result->num_rows;
```

How to display the results with mysqli

```
<?php for ($i = 0; $i < $row_count; $i++) :
        $product = $result->fetch_assoc();
?>
<tr>
    <td><?php echo $product['productID']; ?></td>
    <td><?php echo $product['categoryID']; ?></td>
    <td><?php echo $product['productCode']; ?></td>
    <td><?php echo $product['productName']; ?></td>
    <td><?php echo $product['listPrice']; ?></td>
</tr>
<?php endfor; ?>
```

How to free resources

```
$result->free();    // close the result set
$db->close();       // close the db connection
```

Properties of the mysqli class for checking the result

- `affected_rows`
- `insert_id`
- `error`
- `errno`

How to execute an INSERT statement with mysqli

```
// Execute the statement
$query = "INSERT INTO products
     (categoryID, productCode, productName, listPrice)
          VALUES
     ($category_id, '$code', '$name', $price)";
$success = $db->query($query);

// Check the result
if ($success) {
    $count = $db->affected_rows;
    echo "<p>$count product(s) were added.</p>";
} else {
    $error_message = $db->error;
    echo "<p>An error occurred: $error_message</p>";
}

// Check the product ID that was generated
$product_id = $db->insert_id;
echo "<p>Generated product ID: $product_id</p>";
```

A method of the mysqli class

```
prepare($sql_statement)
```

Four methods of the mysqli_stmt class

```
bind_param($fs, $v1[, $v2]...)
bind_result($v1[, $v2]...)
execute()
fetch()
close()
```

How to execute a prepared statement with mysqli

```
$category_id = 2;
$query = "SELECT productCode, productName, listPrice
          FROM products
          WHERE categoryID = ?";
$statement = $db->prepare($query);
$statement->bind_param("i", $category_id);
$statement->bind_result($code, $name, $listPrice);
$statement->execute();
```

How to display the result set

```
<?php while($statement->fetch()) : ?>
<tr>
    <td><?php echo $code; ?></td>
    <td><?php echo $name; ?></td>
    <td><?php echo $listPrice; ?></td>
</tr>
<?php endwhile; ?>
```

How to close the statement

```
$statement->close()
```

How to use mysqli to execute a prepared statement that adds data

```
$category_id = 2;
$code = 'hofner';
$name = 'Hofner Icon';
$price = 499.99;
$query = "INSERT INTO products
     (categoryID, productCode, productName, listPrice)
          VALUES
     (?, ?, ?, ?)";
$statement = $db->prepare($query);
$statement->bind_param("issd", $category_id, $code,
$name, $price);
$success = $statement->execute();
if ($success) {
    $count = $db->affected_rows;
    echo "<p>$count product(s) were added.</p>";
} else {
    $error_message = $db->error;
    echo "<p>An error occurred: $error_message</p>";
}
$statement->close();
```

The mysqli OO statements compared to its procedural statements

```
$result = $db->query($query);
$result = mysqli_query($db, $query);

$error_message = $db->error;
$error_message = mysqli_error($db);

$row = $result->fetch_assoc();
$row = mysqli_fetch_assoc($result);

$row_count = $result->num_rows;
$row_count = mysqli_num_rows($result);

$count = $db->affected_rows;
$count = mysqli_affected_rows($db);

$result->free();
mysqli_free_result($result);
```

The mysqli OO statements compared to its procedural statements (continued)

```
$statement = $db->prepare($query);
$statement = mysqli_prepare($db, $query);

$statement->bind_param("i", $category_id);
mysqli_bind_param($statement, "i", $category_id);

$success = $statement->execute();
$success = mysqli_execute($statement);

$db->close();
mysqli_close($db);
```

The mysqli model/database.php file

```
<?php
$host = 'localhost';
$username = 'mgs_user';
$password = 'pa55word';
$database = 'my_guitar_shop2';
$db = new mysqli($host, $username, $password, $database);

$error_message = $db->connect_error;
if ($error_message != null) {
    include 'errors/db_error_connect.php';
    exit;
}

function display_db_error($error_message) {
    global $app_path;
    include 'errors/db_error.php';
    exit;
}
?>
```

The mysqli model/category_db.php file

```
<?php
function get_categories() {
    global $db;
    $query = 'SELECT * FROM categories ORDER BY categoryID';
    $result = $db->query($query);
    if ($result == false) {
        display_db_error($db->error);
        exit;
    }
    $categories = array();
    for ($i = 0; $i < $result->num_rows; $i++) {
        $category = $result->fetch_assoc();
        $categories[] = $category;
    }
    $result->free();
    return $categories;
}
```

The mysqli model/product_db.php file

```
<?php
function get_products_by_category($category_id) {
    global $db;
    $query = "SELECT * FROM products
              WHERE categoryID = $category_id";
    $result = $db->query($query);
    if ($result == false) {
        display_db_error($db->error);
    }
    $products = array();
    for ($i = 0; $i < $result->num_rows; $i++) {
        $product = $result->fetch_assoc();
        $products[] = $product;
    }
    $result->free();
    return $products;
}
```

The mysqli model/product_db.php file (continued)

```
function get_product($product_id) {
    global $db;
    $query = "SELECT * FROM products
              WHERE productID = '$product_id'";
    $result = $db->query($query);
    if ($result == false) {
        display_db_error($db->error);
    }
    $product = $result->fetch_assoc();
    return $product;
}
```

The mysqli model/product_db.php file (continued)

```
function add_product($category_id, $code, $name,
        $description, $price, $discount_percent) {
    global $db;
    $query = 'INSERT INTO products
        (categoryID, productCode, productName,
         description, listPrice, discountPercent, dateAdded)
      VALUES
        (?, ?, ?, ?, ?, ?, NOW())';
    $statement = $db->prepare($query);
    if ($statement == false) {
        display_db_error($db->error);
    }
    $statement->bind_param("isssdd", $category_id, $code,
        $name, $description, $price, $discount_percent);
    $success = $statement->execute();
    if ($success) {
        $product_id = $db->insert_id;
        $statement->close();
        return $product_id;
    } else {
        display_db_error($db->error); } }
```

The model/product_db.php file (continued)

```
function update_product($product_id, $code, $name,
    $description, $price, $discount_percent, $category_id)
{

    global $db;
    $query = 'UPDATE Products
          SET categoryID = ?,
              productCode = ?,
              productName = ?,
              description = ?,
              listPrice = ?,
              discountPercent = ?
          WHERE productID = ?';
    $statement = $db->prepare($query);
    if ($statement == false) {
        display_db_error($db->error);
    }
    $statement->bind_param("isssddi",
         $category_id, $code, $name, $description, $price,
         $discount_percent, $product_id);
    $success = $statement->execute();
```

The model/product_db.php file (continued)

```
    if ($success) {
        $count = $db->affected_rows;
        $statement->close();
        return $count;
    } else {
        display_db_error($db->error);
    }
}
```

The mysqli model/product_db.php file (continued)

```
function delete_product($product_id) {
    global $db;
    $query = "DELETE FROM products
              WHERE productID = ?";
    $statement = $db->prepare($query);

    if ($statement == false) {
        display_db_error($db->error);
    }
    $statement->bind_param("i", $product_id);
    $success = $statement->execute();

    if ($success) {
        $count = $db->affected_rows;
        $statement->close();
        return $count;
    } else {
        display_db_error($db->error);
    }
}
?>
```