**ITC 250/CPET 499 Web Systems**

**Nov. 8, 2016**

**Lectures**

**Ch. 11 Managing MySQL Database**

**Topics**

* **PHP Coding Examples (Procedural vs. Object-Oriented)**
  + **Connecting to a MySQL DB**
  + **Handling Connection Errors/Exceptions**
  + **Executing Query:** 
    - **Returning Result Set**
    - **No Result Set**
    - **Prepare Statement**
  + **Integrating User Form Data**
    - **Sanitizing User Data**
  + **Processing the Query Results**
    - **Looping through the Result Set**
    - **Fetching into an Object**
  + **Freeing Resources and Closing Connection**
  + **Using database Transactions**

References

* MySQL Functions, <http://php.net/manual/en/ref.mysql.php>
* PHP Data Objects, <http://php.net/manual/en/book.pdo.php>
* The PDO Class, represents a connection between PHP and a database server, <http://php.net/manual/en/class.pdo.php>

**Connecting to a database: Procedural vs. Object-oriented**

<?php

//Listing11.03.php

// modify these variables for your installation

$host = "localhost";

$database = "bookcrm";

$user = "testuser";

$pass = "mypassword";

$connection = mysqli\_connect($host, $user, $pass, $database);

?>

<?php

// Listing11.04.php

// modify these variables for your installation

$connectionString = "mysql:host=localhost;dbname=bookcrm";

$user = "testuser";

$pass = "mypassword";

$pdo = new PDO($connectionString, $user, $pass);

?>

**Storing Connection Info**

<?php

// Listing11.05.php

define('DBHOST', 'localhost');

define('DBNAME', 'bookcrm');

define('DBUSER', 'testuser');

define('DBPASS', 'mypassword');

?>

<?php

require\_once('Listing10.05.php');//eqivalent for this distribution

//require\_once('protected/config.php'); //original from text

$connection = mysqli\_connect(DBHOST, DBUSER, DBPASS, DBNAME);

?>

**Handling Connection Errors**

<?php

//Listing11.07.php

$connection = mysqli\_connect(DBHOST, DBUSER, DBPASS, DBNAME);

// mysqli\_connect\_error returns string description of the last

// connect error

$error = mysqli\_connect\_error();

if ($error != null) {

$output = "<p>Unable to connect to database<p>" . $error;

// Outputs a message and terminates the current script

exit($output);

}

?>

<?php

//Listing11.08.php

$connection = mysqli\_connect(DBHOST, DBUSER, DBPASS, DBNAME);

// mysqli\_connect\_errno returns the last error code

if ( mysqli\_connect\_errno() ) {

die( mysqli\_connect\_error() ); // die() is equivalent to exit()

}

?>

<?php

//Listing11.09.php

//LISTING 11.9 Handling connection errors with PDO

try {

$connString = "mysql:host=localhost;dbname=bookcrm";

$user = "DBUSER";

$pass = "DBPASS";

$pdo = new PDO($connString, $user, $pass);

//...

}

catch (PDOException $e) {

die( $e->getMessage() );

}

?>

**Setting the PDO Exception Mode**

<?php

//Listing11.10.php

try {

$connString = "mysql:host=localhost;dbname=bookcrm";

$user = "DBUSER";

$pass = "DBPASS";

$pdo = new PDO($connString,$user,$pass);

// useful during initial development and debugging

$pdo->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

//...

}

?>

**Executing the Query**

**Make a Query: Result Set is returned**

<?php

//Listing11.11.php

//Listing 11.11 Executing a SELECT query (mysqli)

$sql = "SELECT \* FROM Categories ORDER BY CategoryName";

// returns a mysqli\_result object

$result = mysqli\_query($connection, $sql);

?>

<?php

//Listing11.12.php

//Listing 11.12 Executing a SELECT query (pdo)

$sql = "SELECT \* FROM Categories ORDER BY CategoryName";

// returns a PDOStatement object

$result = $pdo->query($sql);

?>

**Make A Query (No Data Returned)**

<?php

//Listing11.13.php

//Listing 11.13 Executing a query that doesn't return data (mysqli)

$sql = "UPDATE Categories SET CategoryName='Web' WHERE CategoryName='Business'";

if ( mysqli\_query($connection, $sql) ) {

$count = mysqli\_affected\_rows($connection);

echo "<p>Updated " . $count . " rows</p>";

}

?>

<?php

//Listing 11.14 Executing a query that doesn't return data (PDO)

$sql = "UPDATE Categories SET CategoryName='Web' WHERE CategoryName='Business'";

$count = $pdo->exec($sql);

echo "<p>Updated " . $count . " rows</p>";

?>

**Integrating User Data**

<?php

//Listing.11.15.php

//Listing 11.15 Integrating user input into a query (first attempt)

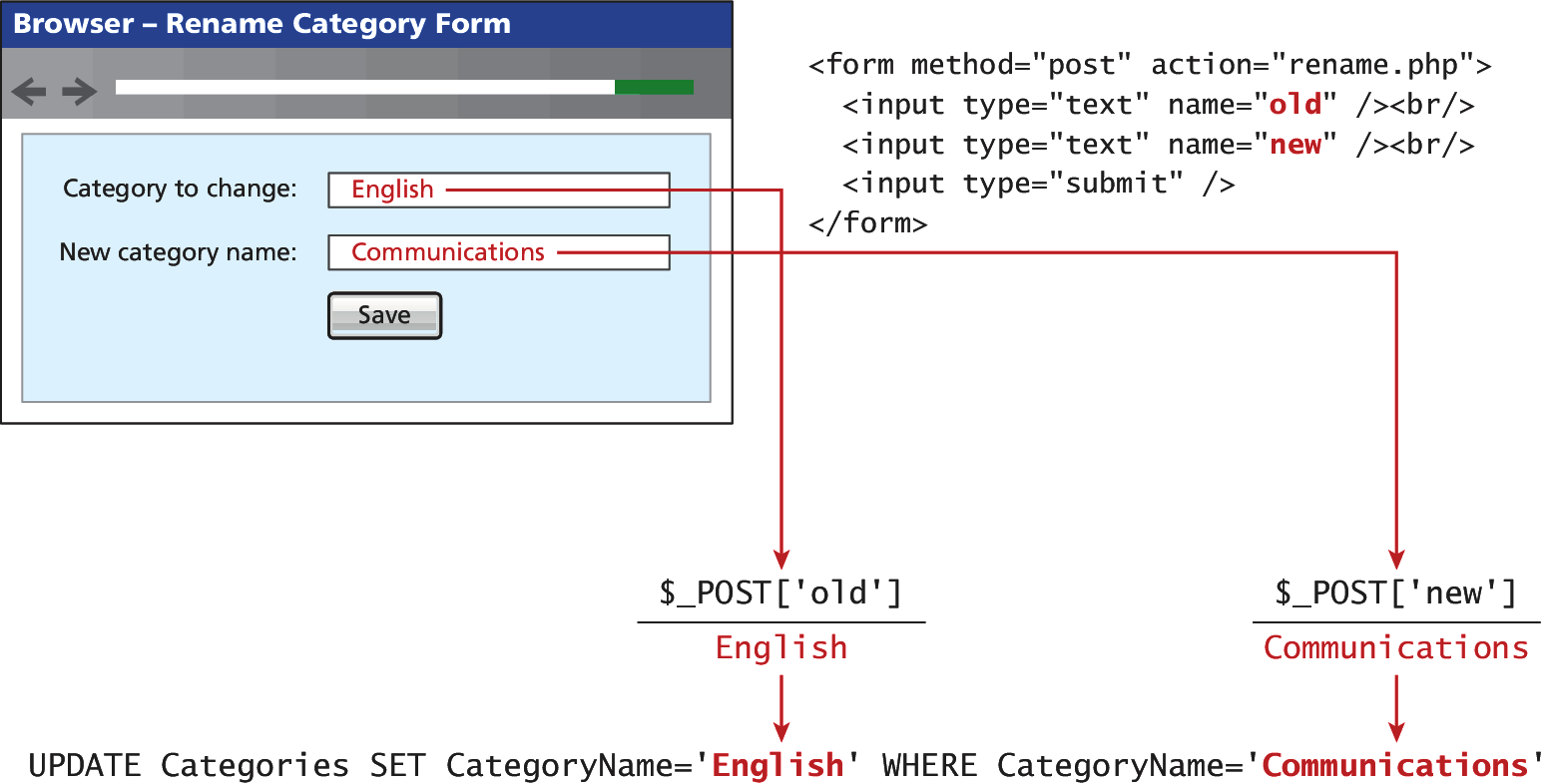
$from = $\_POST['old'];

$to = $\_POST['new'];

$sql = "UPDATE Categories SET CategoryName='$to' WHERE CategoryName='$from'";

$count = $pdo->exec($sql);

?>



**Sanitizing User Data**

PDO::quote, <http://php.net/manual/en/pdo.quote.php>

* Quote a string for use in a query

<?php

//Listing11.16.php

//Listing 11.16 Sanitizing user input before use in an SQL query

$from = $pdo->quote($from);

$to = $pdo->quote($to);

$sql = "UPDATE Categories SET CategoryName=$to WHERE CategoryName=$from";

$count = $pdo->exec($sql);

?>

**Prepared Statements**

<?php

//Listing.11.17.php

//Listing 11.17 Using a prepared statement (mysqli)

// retrieve parameter value from query string

$id = $\_GET['id'];

// construct parameterized query – notice the ? parameter

$sql = "SELECT Title, CopyrightYear FROM Books WHERE ID=?";

// create a prepared statement

if ($statement = mysqli\_prepare($connection, $sql)) {

// Bind parameters s - string, b - blob, i - int, etc

mysqli\_stmt\_bindm($statement, 'i', $id);

// execute query

mysqli\_stmt\_execute($statement);

// learn in next section how to access the returned data

//...

}

?>

<?php

//Listing11.18.php

//Listing 11.18 Using a prepared statement (PDO)

// retrieve parameter value from query string

$id = $\_GET['id'];

/\* method 1 \*/

$sql = "SELECT Title, CopyrightYear FROM Books WHERE ID = ?";

$statement = $pdo->prepare($sql);

$statement->bindValue(1, $id);

$statement->execute();

/\* method 2 \*/

$sql = "SELECT Title, CopyrightYear FROM Books WHERE ID = :id";

$statement = $pdo->prepare($sql);

$statement->bindValue(':id', $id);

$statement->execute();

?>

<?php

//Listing11.19.php

//Listing 11.19 Using named parameters (PDO)

/\* technique 1 - question mark placeholders \*/

$sql = "INSERT INTO books (ISBN10, Title, CopyrightYear, ImprintId,

ProductionStatusId, TrimSize, Description) VALUES

(?,?,?,?,?,?,?)";

$statement = $pdo->prepare($sql);

$statement->bindValue(1, $\_POST['isbn']);

$statement->bindValue(2, $\_POST['title']);

$statement->bindValue(3, $\_POST['year']);

$statement->bindValue(4, $\_POST['imprint']);

$statement->bindValue(4, $\_POST['status']);

$statement->bindValue(6, $\_POST['size']);

$statement->bindValue(7, $\_POST['desc']);

$statement->execute();

/\* technique 2 - named parameters \*/

$sql = "INSERT INTO books (ISBN10, Title, CopyrightYear, ImprintId,

ProductionStatusId, TrimSize, Description) VALUES (:isbn,

:title, :year, :imprint, :status, :size, :desc) ";

$statement = $pdo->prepare($sql);

$statement->bindValue(':isbn', $\_POST['isbn']);

$statement->bindValue(':title', $\_POST['title']);

$statement->bindValue(':year', $\_POST['year']);

$statement->bindValue(':imprint', $\_POST['imprint']);

$statement->bindValue(':status', $\_POST['status']);

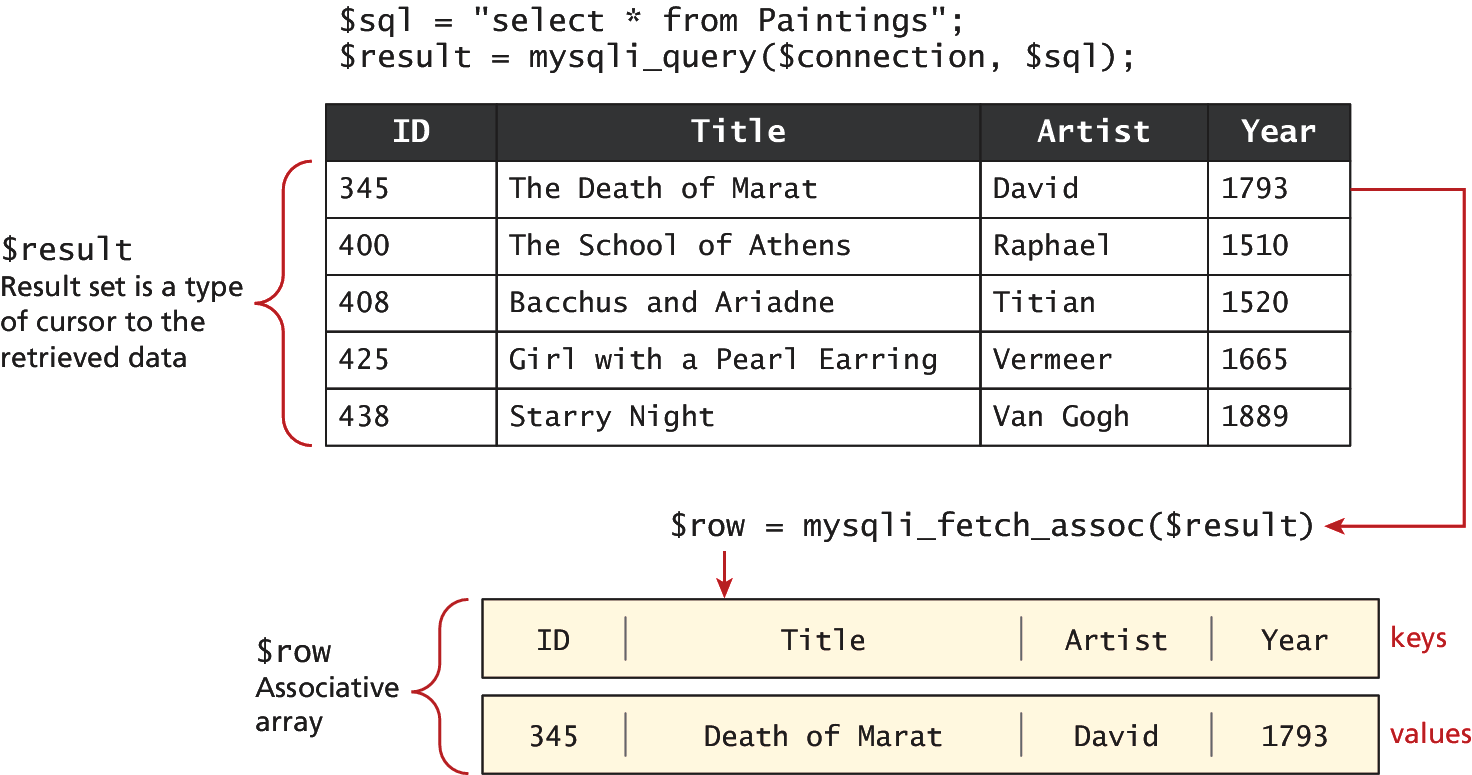
$statement->bindValue(':size', $\_POST['size']);

$statement->bindValue(':desc', $\_POST['desc']);

$statement->execute();

?>

**Processing the Query Results**



<?php

//Listing11.20.php

//Listing 11.20 Looping through the result set (mysqli—not prepared statements)

$sql = "select \* from Categories order by CategoryName";

// run the query

if ($result = mysqli\_query($connection, $sql)) {

// fetch a record from result set into an associative array

while($row = mysqli\_fetch\_assoc($result))

{

// the keys match the field names from the table

echo $row['ID'] . " - " . $row['CategoryName'] ;

echo "<br/>";

}

}

?>

MySQL Funtions (partial list)

* mysqli\_fetch\_all(), mysqli\_fetch\_row()
* mysqli\_fetch\_array(), mysqli\_fectch\_assoc()
* mysqli\_fetch\_field(), mysqli\_fetch\_fields()
* mysqli\_fetch\_object()

**Looping through the Result Set**

<?php

//Listing11.11.21.php

//Listing 11.21 Looping through the result set (mysqli—using prepared statements)

$sql = "SELECT Title, CopyrightYear FROM Books WHERE ID=?";

if ($statement = mysqli\_prepare($connection, $sql)) {

mysqli\_stmt\_bindm($statement, 'i', $id);

mysqli\_stmt\_execute($statement);

// bind result variables

mysqli\_stmt\_bind\_result($statement, $title, $year);

// loop through the data

while (mysqli\_stmt\_fetch($statement)) {

echo $title . '-' . $year . '<br/>';

}

}

?>

<?php

//Listing.11.22.php

//Listing 11.22 Looping through the result set (PDO)

$sql = "select \* from Categories order by CategoryName";

$result = $pdo->query($sql);

while ( $row = $result->fetch() ) {

echo $row['ID'] . " - " . $row['CategoryName'] . "<br/>";

}

?>

**Fetching into an Object**

class Book{

public $id;

public $title;

public $copyrightYear;

public $description;

}

<?php

//Listing11.23.php

//Listing 11.23 Populating an object from a result set (PDO)

$id = $\_GET['id'];

$sql = "SELECT id, title, copyrightYear, description FROM Books WHERE id= ?";

$statement = $pdo->prepare($sql);

$statement->bindValue(1, $id);

$statement->execute();

$b = $statement->fetchObject('Book');

echo 'ID: ' . $b->id . '<br/>';

echo 'Title: ' . $b->title . '<br/>';

echo 'Year: ' . $b->copyrightYear . '<br/>';

echo 'Description: ' . $b->description . '<br/>';

?>

<?php

//Listing11.24.php

//Listing 11.24 Letting an object populate itself from a result set

class Book {

public $id;

public $title;

public $copyrightYear;

public $description;

function \_\_construct($record)

{

// the references to the field names in associative array must

// match the case in the table

$this->id = $record['ID'];

$this->title = $record['Title'];

$this->copyrightYear = $record['CopyrightYear'];

$this->description = $record['Description'];

}

}

//...

// in some other page or class

$statement->execute();

// using the Book class

$b = new Book($statement->fetch());

echo 'ID: ' . $b->id . '<br/>';

echo 'Title: ' . $b->title . '<br/>';

echo 'Copyright Year: ' . $b->copyrightYear . '<br/>';

?>

**Freeing Resources and Closing Connection**

<?php

//Listing 11.25 Closing the connection

// **mysqli approach**

$connection = mysqli\_connect($host, $user, $pass, $database);

//...

// release the memory used by the result set. This is necessary if

// you are going to run another query on this connection

mysqli\_free\_result($result);

//...

// close the database connection

mysqli\_close($connection);

// **PDO approach**

$pdo = new PDO($connString,$user,$pass);

//...

// closes connection and frees the resources used by the PDO object

$pdo = null;

?>

**Using Transactions**

* Unnecessary when retrieving data
* Should be used for database writes

<?php

//Listing11.26.php

//Listing 11.26 Using transactions (mysqi extension)

$connection = mysqli\_connect($host, $user, $pass, $database);

//...

/\* set autocommit to off. If autocommit is on, then mysql will

commit (i.e., make the data change permanent) each command after

it is executed \*/

mysqli\_autocommit($connection, FALSE);

/\* insert some values \*/

$result1 = mysqli\_query($connection,"INSERT INTO Categories (CategoryName) VALUES ('Philosophy')");

$result2 = mysqli\_query($connection,"INSERT INTO Categories (CategoryName) VALUES ('Art')");

if ($result1 && $result2) {

/\* commit transaction \*/

mysqli\_commit($connection);

}

else {

/\* rollback transaction \*/

mysqli\_rollback($connection);

}

?>

<?php

//Listing11.27.php

//Listing 11.27 Using transactions (PDO)

$pdo = new PDO($connString,$user,$pass);

// turn on exceptions so that exception is thrown if error occurs

$pdo->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

//...

try {

// begin a transaction

$pdo->beginTransaction();

// a set of queries: if one fails, an exception will be thrown

$pdo->query("INSERT INTO Categories (CategoryName) VALUES ('Philosophy')");

$pdo->query("INSERT INTO Categories (CategoryName) VALUES ('Art')");

// if we arrive here, it means that no exception was thrown

// which means no query has failed, so we can commit the

// transaction

$pdo->commit();

} catch (Exception $e) {

// we must rollback the transaction since an error occurred

// with insert

$pdo->rollback();

}

?>