CSCI 466/566

SQL via PHP

Jon Lehuta

Northern Illinois University

November 2, 2015

Northern Illinois University

SQL via PHP

Introduction

Using PDO

Northern Illinois University

Why SQL via PHP?

- Can provide an interface for the user that does not require them to worry about database design specifics.
- Although there are other ways to provide this interface, the web-based interface is very common, and PHP is a common and relatively easy way of making it work.

Application Programming Interface

In order for our PHP application to interface with our DBMS, we will need to use an appropriate API (Application Programming Interface) An API is the set of function calls and other resources that are provided to allow you to interface with a given application via your program code.

Which API?

Even for a given DBMS, there can be many API's present. PHP has been around for a while, so many things have evolved and died out. Many of the API's still work. Some work but are considered deprecated, and others are no longer supported at all. In this class, we will be working with the PDO library.

PDO Library

The PHP Data Objects (PDO) library is an object oriented API to connect PHP to SQL servers. It allows you to use a common interface to interact with many different DBMS programs.
It supports most of the popular relational DBMSes, including MySQL, Postgresql, and SQLite, in a fairly transparent way, so it is more portable than using the other, DBMS specific API's.
Note: Because PDO is object oriented, it requires at least version 5 of PHP. If you need to use a lower version, you'll need to look into the other API's available.

Getting Started With PDO

Although, once properly initialized, PDO should function the same for any DBMS, you need to properly initialize it by telling it which type of server you are connecting to. To do this, you need to make a DSN string.

DBMS	DSN Format
MySQL	`mysql:host=HOSTNAME;dbname=DBNAME`
Postgresql	`pgsql:host=HOSTNAME;dbname=DBNAME`
SQLite 3	`sqlite:PATHTODB`
SQLite 2	`sqlite2:PATHTODB`

Northern Illinois University

Initializing PDO

Once you've chosen the DBMS you'll be using and you've chosen an appropriate DSN string, you can use that DSN to construct an instance of the PDO class. This object will be used to communicate with your database.

```php
<?php
try { // if something goes wrong, an exception is thrown
  $dsn = "mysql:host=courses;dbname=z123456";
  $pdo = new PDO($dsn, $username, $password);
}
catch(PDOexception $e) { // handle that exception
  echo "Connection to database failed: " . $e->getMessage();
}
?>
```

Using PDO to Talk to DBMS

The PDO library provides three basic ways of running queries for a database, once connected:

- the `exec()` function is used to execute an SQL query that does not return a result (`INSERT`, `UPDATE`, etc.)
- the `query()` function is used to execute an SQL query that will return a result (`SELECT`)
- the `prepare()` function should be used when constructing a query from user input.

Using exec()

exec() is used to run a query that does not return any results. Instead of returning a result set, it will tell you how many rows were affected by your query.

```php
<?php
  // Three examples follow.
  $n = $pdo->exec("INSERT INTO Students (FName) VALUES ('Victor');");

  $n = $pdo->exec("UPDATE Students SET LName='Husky' WHERE FName='Victor';");

  $n = $pdo->exec("DELETE FROM OldJunk;");
?>
```

Northern Illinois University

Using query()

query() is used to run a query that does return a result. The result set is returned as a PDOStatement object.

```php
<?php
  # Defining $sql as the query you'd like to run; here's one from classicmodels
  $sql = "SELECT phone FROM Customer;";

  # Run the query - the results are stored into the $result object on success
  $result = $pdo->query($sql);
?>
```

Using prepare()

The third option is to use the `prepare()` command. This is useful for situations where the same query is run multiple times in the same script, and can also help you to avoid some SQL Injection issues. Once `prepare()` succeeds, you run the query with `execute()`
You can use a colon before a value name in your query to denote where the `execute` statement will insert the value of the given name:

```php
<?php
  # Notice that we use :color below in the prepare
  $sql = 'SELECT name, color, calories
            FROM fruit
            WHERE calories < :calories AND color = :color';
  $prepared = $pdo->prepare($sql, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
  # The value associated with the :color key will be inserted into the SQL when exec
  $result = $prepared->execute(array(':calories' => 150, ':color' => 'red'));
?>
```

Using prepare()

It is also possible to use a ? in your query as a positional parameter.

```php
<?php
  $ Execute a prepared statement by passing an array of values
  $prepared = $pdo->prepare('SELECT name, color, calories
                             FROM fruit
                             WHERE calories < ? AND color = ?');

  # Here we execute the query twice with different parameters.
  # The ?'s will be replaced with the values in the array specified,
  # in the order they are specified.
  $prepared->execute(array(150, 'red'));
  $red = $prepared->fetchAll();
  $prepared->execute(array(175, 'yellow'));
  $yellow = $prepared->fetchAll();
?>
```

Dealing with result sets

Once you have a result set from `query` you can use the `fetch` or `fetchAll` methods on it.
If you would like to work on one row at a time, as if using the `mysql_fetch_array` function from the original MySQL API, use `fetch`.

```
<?php
  # FETCH_BOTH means that you will get both position indices and the column names
  # as keys in the array returned
  $row = $result->fetch(PDO::FETCH_BOTH);

  # this returns all of the rows at once in a two-dimensional array
  $allrows = $result->fetchAll();
?>
```

More on PDO

This was just a brief introduction to PDO. You should be able to use what you find in these slides to do most things, but if you are looking to accomplish something more complicated, or would like to learn more about what is available, you can check out the PDO reference on the PHP site at the following URL:

`http://php.net/manual/en/intro.pdo.php`