An Introduction to PHP Data Objects

A Better Way to Interact with Your Database

by

Rusty Keele
This Presentation Is For People Who...

- Want to learn the basics of PHP Data Objects (PDO)
- Want to start using PDO in their code
- Have a basic knowledge of PHP and SQL
What We’ll Cover

- My story
- An overview of PDO
- Examples and real PHP code
- Wrap up
My Story
My History with Programming

• Started on a Commodore 64 – in 1982!

• B.A. in Computer Science from Weber State University

• 9 years as a full time programmer – mostly PHP

• Worked for Utah Education Network the last 4 years
How I Got Started With PDO

• What I inherited at UEN
• The mysql_x() revelation
• Switching from MySQL to PostgreSQL ?!
• 2 birds with 1 stone
PDO: An Overview
What Is PDO?

- “The PHP Data Objects (PDO) extension defines a lightweight, consistent interface for accessing databases in PHP.” - php.net

- Is: a data-access abstraction layer
- Is not: a database abstraction layer
- Is not: an object relation mapping (ORM) system
- Does not: rewrite SQL
- Does not: emulate missing features
Why Use PDO?

- A future change of database
- Prepared statements
- Rewriting code
- Object oriented programming
- You’re writing your own framework! ;-}
First Steps

- PDO is enabled by default as of PHP 5.1.0
The Main PDO Classes

- **PDO**: Represents a connection between PHP and a database server.

- **PDOStatement**: Represents a prepared statement and, after the statement is executed, an associated result set.

- **PDOException**: Represents an error raised by PDO.
Other Important Ideas

• PDO drivers
  • drivers implement the PDO interface for specific databases
  • MySQL, PostgreSQL, MS SQL Server, Oracle...

• Prepared statements
  • a kind of compiled template for the SQL that an application runs
  • can be customized using variable parameters
  • prepare once – execute multiple times
  • don't need to be quoted – which helps defend against SQL injection!
Using PDO
Connecting To A Database

- Connect

```php
$dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass);
```

- Close

```php
$dbh = null;
```

- Handle errors

```php
try {
    $dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass);
    foreach($dbh->query('SELECT * from FOO') as $row) {
        print_r($row);
    }
    $dbh = null;
} catch (PDOException $e) {
    print "Error: " . $e->getMessage() . "<br/>";
    die();
}
```
Prepared Statements: Reading Data

• Using named placeholders

```php
$sth = $dbh->prepare('SELECT name, color, calories
                     FROM fruit
                     WHERE calories < :calories AND colour = :color);
$sth->bindParam(':calories', 150);
$sth->bindParam(':color', 'red');
$sth->execute();
```

• Using question mark placeholders, variables, data type and length

```php
$calories = 150;
$colour = 'red';
$sth = $dbh->prepare('SELECT name, color, calories
                     FROM fruit
                     WHERE calories < ? AND color = ?');
$sth->bindParam(1, $calories, PDO::PARAM_INT);
$sth->bindParam(2, $color, PDO::PARAM_STR, 12);
$sth->execute();
```
Prepared Statements: Inserting Data

- Binding an array when executing and getting last insert ID

```php
$bind_arr = array();
$bind_arr['first_name'] = $first_name;
$bind_arr['last_name'] = $last_name;
$bind_arr['email'] = $email;

$sql = "INSERT INTO `people` (`id`, `first_name`, `last_name`, `email`) VALUES (NULL, :first_name, :last_name, :email);"

$sth = $dbConn->prepare($sql);
$sth->execute($bind_arr);
$attendee_id = $dbConn->lastInsertId();
```
Deleting Data

• Using exec( ) to get number of rows affected

```php
$count = $dbh->exec("DELETE FROM fruit WHERE color = 'red'"elsen)
print("Deleted $count rows. \n");
```

• Differences between execute( ) and exec( )
  • exec( )
    • A method of the PDO class
    • Executes an SQL statement and returns the number of affected rows
  • execute( )
    • A method of the PDOStatement class
    • Executes a prepared statement and returns TRUE or FALSE
Putting It All Together

```php
/**
 * Gets a list of today's meetings from the database
 */
public function getTodaysMeetings() {
    $retArr = array();
    $DbConnection = new DbConnection();
    $dbConn = $DbConnection->connect_pdo();

    $today_dt = new DateTime();
    $today_dt->setTime(0,0,1);
    $bind_arr['start_time'] = date_format($today_dt, 'Y-m-d H:i:s'); // '2014-03-21 00:00:01'
    $today_dt->setTime(23,59,59);
    $bind_arr['end_time'] = date_format($today_dt, 'Y-m-d H:i:s');

    $sth = $dbConn->prepare("SELECT *
                         FROM ivoarnt_sched
                         WHERE start_time > :start_time
                         AND end_time < :end_time
                         ORDER BY start_time ASC, end_time ASC;");
    $sth->execute($bind_arr); // bind the array and execute the prepared statement...
    $retArr = $sth->fetchAll(PDO::FETCH_ASSOC);
    $dbConn = null; // close the DB connection...
return $retArr;
}
```
Gotchas!

• Be careful when using “LIKE %keyword%”

```php
$bind_arr = array();
$bind_arr[] = "%%$keyword%%";
$bind_arr[] = "%%$keyword%%"

$sql = "SELECT *
        FROM `lrn_res`
        WHERE `name` LIKE ?
        OR `description` LIKE ?; ";

$sth = $dbConn->prepare($sql);
$sth->execute($bind_arr);
$results = $sth->fetchAll(PDO::FETCH_ASSOC);
if($results) {
    foreach($results as $row) {
        $arr[] = $row;
    }
}
```
Support For Transactions

- Only works with databases that support transactions
- Transactions are typically implemented by "saving-up" your batch of changes to be applied all at once
- Can roll back on errors

```php
try {
    $dbh->beginTransaction();
    $dbh->exec("insert into staff (id, first, last) values (23, 'Joe', 'Bloggs')");
    $dbh->exec("insert into salarychange (id, amount, changedate)"
                . "values (23, 50000, NOW())");
    $dbh->commit();
}
catch (Exception $e) {
    $dbh->rollBack();
    echo "Failed: " . $e->getMessage();
}
```
Changing Your Database

• In theory, you can just switch the Data Source Name in the connection string
  • From: mysql:host=localhost;dbname=testdb;user=un;password=pw
  • To: pgsql:host=localhost;dbname=testdb;user=un;password=pw

• Watch out for non-standard SQL!
  • For example: MySQL’s “replace into”
PDO And Frameworks

- Not really needed because of abstraction layers
- Especially if using ORM
- CodeIgniter has a PDO driver option in version 3.0
- Check the documentation for your framework: Laravel, CakePHP, Zend 2
Wrapping Things Up
The Rest Of The Story

• Spent a year working on this at UEN (the MySQL to PDO part)... one file at a time!

• Now I have better, non-deprecated, more secure, uncoupled, object oriented code for database connections – whew!

• Next: change the back end database to PostgreSQL!
Where To Learn More About PDO


• A good intro article: http://www.creativebloq.com/design/using-php-data-objects-1133026

• Books (on Amazon.com)
  • Learning PHP Data Objects: A Beginner’s Guide to PHP Data Objects, Database Connection Abstraction Library for PHP 5
  • PHP Data Objects Quick Start
Picture Credits

- All photos from [www.flickr.com](http://www.flickr.com)
- All photos under Creative Commons License
- Wolf by alexandre alacchi, Wolf Park by Debs, Wolf looking at me by Tambako The Jaguar, Pretty standing wolf by Tambako The Jaguar, Wolves fighting over food by Tambako The Jaguar, Wolf by Jimmy Álvarez, Wolf by Cloudtail, Black Wolf by numb - El Condor, C'est un garçon by peupleloup
- Commodore 64 by Steve Petrucelli
- Darn Itch by Brave Heart
- cartwheel by Julie
Feedback And Questions

• Please rate my presentation and give feedback
  • https://joind.in/13933

• Where to get my slides
  • http://c64sets.com/slides/

• Questions? Comments?