# Database Driven Websites Using PHP with Informix

February 12, 2013

**Thomas Beebe**

***Advanced DataTools Corp***

(tom@advancedatatools.com)

Advanced DataTools

# Tom Beebe

Tom is a Senior Database Consultant and has been with Advanced DataTools for over 10 years. He has been working with Informix since college with a long time fondness for open source languages. Tom is the lead consultant for Networking, Unix System Administration and Web Development needs. Currently, he is the Project Manager and lead developer on a variety of Web Development projects.

Contact Info:
**tom@advancedatatools.com**
**www.advancedatatools.com**
**703-256-0267 x 106**

Advanced DataTools

# Introduction To PHP

- Stands For PHP: Hypertext Preprocessor
- Started in 1995 by Rasmus Lerdorf as perl scripts
- Andi Gutmans and Zeev Suraski launched PHP3 in 1997 to replace the dated PHP/FI
- Started in 1995 as a set of perl scripts
- Rewritten in 1997 as C
- PHP3 came in 1998
- PHP4 in May 2000
- PHP5 in July 2004
- Provides a very flexible, expendable application development environment, free to use for any purpose.
- Large collection of modules and extensions available (mostly free of charge)

# Why Use PHP?

- Fast development time
- Flexible coding styles
- Widely supported
- Extensive support options
- Free
- Excellent performance

# Run Methods

- CLI – command line
- Web module:
  - Apache (1/2/2.2)
  - IIS
  - Websphere
  - Iplanet
  - Lighttpd
  - CGI
- Etc, Etc

# Installation Notes

- Most Linux systems have it in their package repositories
- Windows installers available
- Will compile on most any UNIX system.
- Will compile from source on Windows.
- Configure is the standard way, use --help

# PECL

- Online package repository.
- Originally on pear, now moving to pecl
- Phpize.
- PDO Informix Drivers
- Excel and Word support
- Clucene
- http://pecl.php.net/

# Structure of a PHP File

```
<?  <?php
echo 'hello world';
?>
```

**Also:**

```
<html><body>Hello World<br>
<? echo $hello_world; ?>
</body></html>
```

**Can also do:** <?=$hello_world?>

# Phpize

Useful once a system is set up.

Can be run from the command line as well.

Used to configure the build system for modules

# PDO

- Standardized code and connection methods
- ODBC can be troublesome
- The ifx_ driver is not overly stable and is buggy
- Open source development methodology
- IBM supported pdo_informix and pdo_ibm drivers
- Proper error handling

# PDO_IBM vs PDO_Informix

- PDO_IBM is the official direction that IBM is going in
- PDO_Informix is used by OAT
- PDO_IBM does not work with lvarchar nor some of the Informix specific functions
- PDO_Informix has not been updated in several years (last update 11/10/08)
- Do not co-exist on the same system.

# Installing

- Two ways to install:
  - extensions (php.ini)
  - compiled in ($PHP_SRC/php-x.x.x/ext/pdo_informix)

# Compiling into PHP

- Download the latest pdo_informix driver
- Extract it to php_source/ext [it will be PDO_INFORMIX-1.x]
- Rename it to pdo_informix
- cd ..
- ./buildconf –force [this will rebuild configure]
- ./configure –help | grep informix (make sure it is there)
- ./configure –args --with-pdo-informix=$INFORMIXDIR

# Installing From Pecl

pecl –install pdo_informix

- Add:
  - Extension=pdo_informix.so to php.ini

  This will only work if everything is clean.

# Manually Build Module

- Download the latest PDO-Informix from the pecl website
- Unpack
- Run 'phpize'
- ./configure; make; make install
- Edit configure file if needed.
- Add extension=pdo_informix.so
- (pdo_informix.dll if on windows)

# Dealing With Windows

Does not come standard with PHP.

Options:

- Use ODBC
- Use OAT
- Find a pre-built dll on a similar system.

# PDO Connect

- This will create a new PDO object named $dbh
- $dbh = new PDO("informix:DSN=test_odbc”,  $user, $password);

Or:

```
$dbh = new
    PDO("informix:host=localhost;service=1516;database=stores_demo;
    server=ifxserver;protocol=onipcshm;EnableScrollableCursors=1",
    $user,$password);
```

# Connection Paramaters

- $dbh->setAttribute($ATTR,$VALUE);
- PDO::ATTR_CASE:
  - PDO::CASE_LOWER
  - PDO::CASE_UPPER
  - PDO::CASE_NATURAL
- PDO::ATTR_ERRMODE:
  - PDO::ERRMODE_SILENT
  - PDO::ERRMODE_WARNING
  - PDO::ERRMODE_EXCEPTION
- PDO::ATTR_STRINGIFY_FETCHES

# PDO Basic Query

```
$sqlline = “select count(*) from foo”;
$dbh->query($sqlline);
Or
$dbh->prepare($sqlline);
$dbh->execute();
```

# PDO Fetching Data

- To get a statement handle set a variable to an executed query
- $sth = $dbh->query($sqlline);
- To return data do $row = $sth->fetch(PDO::FETCH_ROW);
- PDO::FETCH_ASSOC
- PDO::FETCH_BOTH
- PDO::FETCH_NUM
- PDO::FETCH_OBJ
- PDO::FETCH_LAZY

# PDO Bound Parameters

- These make life far easier
- All data will be inserted properly, no worries about cleaning the data or incorrectly placed quotes
- Prevents SQL injection attacks.
- Properly handles newlines
- Can be bound by placement or by parameter name
- Faster

# Binding on Selects

- This is required for reading blobs from the database
- $stmt = $db->prepare("select cat_desc from catalog”);
- $stmt->execute();
- $stmt->bindColumn(1, $lob, PDO::PARAM_STR);
- $stmt->fetch(PDO::FETCH_BOUND);

# Binding Paramaters

- LOB needs to be specified, the others are optional but can help maintain proper data types
- PDO::PARAM_LOB – Data Blobs
- PDO::PARAM_STR – Char/Varchar/Text
- PDO::PARAM_BOOL – Bool
- PDO::PARAM_INT - Integer

# PDO Bound Parameters

- $sqlline = “update users set last_IP = ? where user_id = ? and password = ?”;
- $sth = $dbh->prepare($sqlline);
- $vars = array($ip, $username, $passwd);
- $sth->execute($vars);

# PDO Bound Parameters

- $ip = $_SERVER['REMOTE_ADDR'];
- $username = 'test_user';
- $password = 'weakpass';
- $sqlline = “update users set last_ip = :IP where username = :user and password = :pass”;
- $sth = $dbh->prepare($sqlline);
- $sth->bindParam(':IP', $ip, PDO::PARAM_INT);
- $sth->bindParam(':user', $username);
- $sth->bindParam(':pass', $password, PDO::PARM_STR, 20);
- $sth->execute();

# PDO Fetch (cont.)

- fetch()
- fetchAll()
- fetchColumn($col_num)

# Error Handling

- PHP 5 introduced try/catch which can be a lifesaver.
- try { sql stuff here...
- } catch (PDOException $e) {
- print "Error!: " . $e->getMessage();
- die();
- }

# Error Handling (cont)

- $dbh->errorInfo();
  - Returns an array of the last statement executed
  - 0: Sqlstate Error Code
  - 1: Driver specific error number
  - 2: Driver error message

- Exception handling variable commands
  - $e->getMessage() = Error message
  - $e->getLine() = Line Number
  - $e->getFile() = File that caused the error
  - $e->getCode() = Code the caused the error
  - $e->getTrace() = Array of the error trace

# Links

- http://www.php.net
- http://framework.zend.com/
- http://www.zend.com/
- http://www.iiug.org/opensource
- http://pecl.php.net/package/PDO_INFORMIX
- http://pecl.php.net/package/PDO_IBM
- http://www.openadmintool.org

# Database Driven Websites Using PHP with Informix

February 12, 2013

**Thomas Beebe**

***Advanced DataTools Corp***

(tom@advancedatatools.com)

Advanced DataTools