Getting Up and Running With PHP with Informix

March 18th 2014

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 and is currently the lead consultant for Networking, Unix System Administration and Web Development. Tom is 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

About This Webcast Series

Four Webcasts:

- March 18th – Getting started
- April 22nd – Data structures, objects, functions and database access.
- May 20th – Introduction to web focused programming. Creating simple and advanced forms
- June 17th – Putting it all together, assorted advanced topics.

About 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
- Fast performance
- Scalable

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

Structure of a PHP File

Begin Script:

- `<?  <?php <?php3`

Command:

- `echo 'hello world';`

End Script:

- `?>`

Also:

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

Can also do:

- `<?=$hello_world?>`

Basic Script

```
<?php
print “hello world”;
?>

<?php
for ($x=0;$x<5;$x++){
    print “X is $x\n”;
    }
    ?>
```

phpinfo();

- Standard function that will give you extensive information about your php configuration
- Web based output
- Do not put on a public website

Variables

```
$x = 5;
$x = “Testing”;
print “X is $x\n”;
```

X is Testing

```
$x = 3;
$x = $x * 2;
print “X is $x\n”;
```

X is 6

Arrays

Arrays can be numeric:

```
$var[0] = 'apple';
$var[1] = 'pear';
```

Associative:

```
$var['first'] = 'apple';
$var['second'] = 'pear';
```

Multidimensional:

```
$var[0]['first'] = 'apple';
$var[1]['first'] = 'pear';
```

Objects

Used very commonly in PHP5

A variable can be defined as an object, giving it access to all of the methods of that object.

```
$foo = new object();
$foo->run_function();
$foo->var = 5;
print “The value of obj foo, variable var is: “. $foo->var .”\n”;
```

php.ini

Controller file for how PHP executes

Adjust memory, permissions, paths, error levels, modules all from a single file.

Some Linux distributions split it into a conf.d directory

Run phpinfo(); to get the path, the path can be also be configured at runtime.

PECL

Online package repository.

Originally on pear, now moving to pecl

Some Linux distributions require php-devel to support it.

Some commonly used modules:

- PDO Informix Drivers
- Excel and Word support
- Clucene
- Graphing
- Script Caching

Can be found at: http://pecl.php.net/

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

- Both drivers are widely available
- PDO_Informix is used by OAT
- PDO_IBM runs over DRDA (v11 or later only) also some of the Informix specific functions do not work.
- PDO_Informix was last updated June 2013
- Make sure you are on the latest version.
- Sometimes they do not play nicely on the same system.

Installing

- Three ways to install:
  - Compiled into the engine
  - Pecl
  - Compiled as a module

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
- Add extension=pdo_informix.so
- (pdo_informix.dll if on windows)
- Some systems will have a bug on configure, line 4669
- If php_pdo_driver is not in the pre-defined location, add in:

```
elif test -f /usr/include/php5/ext/pdo/php_pdo_driver.h; then
  pdo_inc_path=/usr/include/php5/ext
```

Before 'else'

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=nfl;server=ifxl
appy;protocol=onipcshm;", "informix","inf0rmix");

Setting database handle attributes:

$dbh-
>setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTI
ON);

Connection Paramaters

- Database Handle Options
- Set Via: $dbh->setAttribute($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 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

- $ip = $_SERVER['REMOTE_ADDR'];
- $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 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);
- Different data types to return the data:
  - PDO::FETCH_ASSOC
    - $row['first_name'] = 'Tom';
  - PDO::FETCH_BOTH
    - $row['first_name'] = 'Tom';
    - $row[0] = 'Tom';
  - PDO::FETCH_NUM
    - $row[0] = 'Tom';
  - PDO::FETCH_OBJ
    - $row->first_name = 'Tom';
  - PDO::FETCH_LAZY
    - All of the above

PDO Fetch (cont.)

- Fetch by taking the statement handle and calling one of the following functions:
  - fetch()
    - Returns one single row, moves cursor to the next row.
  - fetchAll()
    - Returns all rows as a large array.
  - fetchColumn($col_num)
    - Fetches the value of a single column, moves the cursor to the next row.

Error Handling

- PHP 5 introduced try/catch which can be a lifesaver.

```
try {
        do_something();
} catch (PDOException $e) {
  print "Error!: " . $e->getMessage() . "<br/>";
  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

Getting Up and Running With PHP with Informix

March 18th 2014

Thomas Beebe

Advanced DataTools Corp

(tom@advancedatatools.com)

Advanced DataTools