IBM i and Zend PHP for old folks – 02 getting a class connection

louwilkinson1@gmail.com

Setup:

Again, this assumes you're pretty experienced with IBM midrange computers...and just starting with PHP. It also assumes you finished the last .pdf and have a **connectiontest02.php** out there from our last get together.

Also, and it can't be stressed enough, we're blowing past a lot of standard best practice stuff here. I'm trying to show things using the smallest amount of code and not clutter the scripts up with error checking etc. But you should. One of the reasons to be even using PHP is because there are so many resources out there to help you get started.

Today's topic...turning what you intuitively understand...a functional process with external functions into a class and using “object oriented” methods to do something. The simplest most basic overview. There's a ton of info on the internet and in your local bookstore on oo stuff...I just want to show you how one simple thing can be converted to get you started.

When we last met, we had an external set of functions called **ConnectFunctions.php** and a little test script to test it.

What we're going to do is to take **ConnectFunctions.php** and turn it in a **class**, an “object”, and then we'll set up another test script (**connectiontest03.php**) to use it.

So let's go! This is what we have now:

```
<?php

function dbsetup(){
      if (!defined('DATABASE'))
            define('DATABASE', "yourdb");
      if (!defined('USER'))
            define('USER', "qpgmr");
      if (!defined('PASSWORD'))
            define('PASSWORD',"qpgmrpassword");
};

function Connect(){
      dbsetup();
      return  db2_connect(DATABASE, USER, PASSWORD);
};

function Close($connection){
      dbsetup();
      return db2_close($connection);
}
?>
```

I'm not going to go over that again...I'm simply going to remind you that we have some constants for your setup and two functions, Connect() and Close().

And I'm not going to go into the depth of object oriented programming (ha! as if I could!). I'm just going to say this:

1) you know that you have a “thing” that's a connection from your PHP script to your database...

2) and a “thing” is an object.

So fire up your browser again and let's create a new file in **/www/zendsvr/secure** called **connection.php.** Then type this in, putting in your db name and your qpgmr password as we discussed before, and save it.

```
<?php

// “Class” says that this is an object and I'm calling this “type”
//  of object an iSeries.

Class iSeries {

	// the constants we've been using

	const DATABASE = "yourdb";
	const USER = "qpgmr";
	const PASSWORD = "qpgmrpassword";


	// a public variable for our connection

	public $conn;

	// first function – look kind of familiar?

	public function connect() {
		$this->conn = db2_connect(self::DATABASE,
					self::USER,
					self::PASSWORD);
		return $this->conn;
		}

	//  close function...  $this->conn is kind of object oriented-ese for
	//  “the one i'm talking to you about...”  you'll see a lot of this->

	public function close() {
			return db2_close($this->conn);
			}
	}
?>
```

No, you're not supposed to understand all that. What you ARE supposed to glean is that there are two functions that are analogous to the two we were already using...and there's some oo type jargon....self:: and this->...but that doesn't prohibit you from looking it over and seeing that it makes sense...that it's not THAT different, in concept, from what you know well.

On to our test script which will actually USE our object.

Open the editor and create **connectiontest03.php** in **www/zendsvr/htdocs/testing**.

I'm going to run through it with english comments behind the // and then with object-ese comments underneath the english.

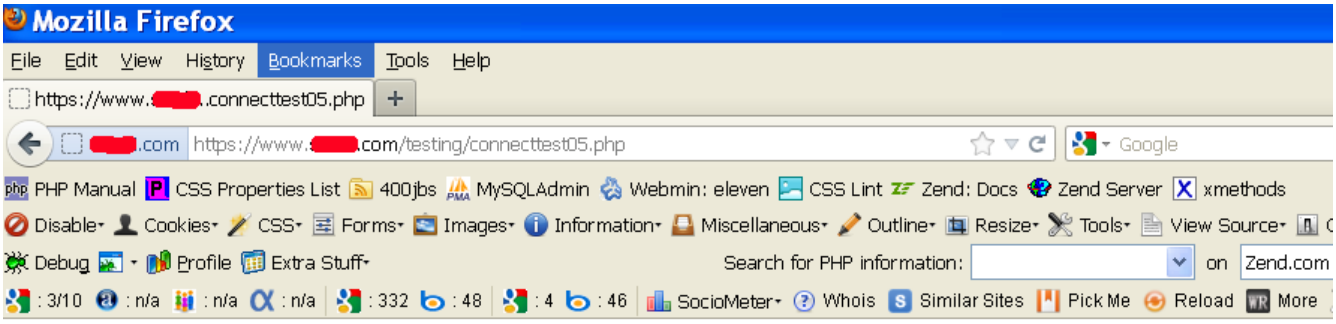
```
<?php

        // you know this...from our last get together.

        include_once "../../secure/connection.php";

        // i'd like to create a new thing (that will be a db connection) and I want to call it Scarlet
        // i'd like to instantiate an iSeries object named Scarlet

        $Scarlet = new iSeries;

        // i'd like to actually connect to scarlet, please
        // i'd like to invoke the connect() method for Scarlet

        $conn = $Scarlet->connect();

        if (!$conn) {
              echo "<br />\n no connection!!!";
              }
        else {
              echo "<br />\n Connection OK ";
              }

        // i'd like to close the connection to scarlet.
        // i'd like to invoke the close() method for Scarlet

        $rc = $Scarlet->close();

        if ($rc) {
              echo "<br />\n Connection was successfully closed.";
        }
        else {
              print "<br />\n nope...not yet.";
        }

?>
```

see how similar it is to our last script? Yes, the → and all that are different...but, again, you can read this stuff...you can follow it.

Already the neighborhood kids are looking you over with newfound respect. The 20somethings in your shop are beginning to give each other the eye. You know what I mean.

So, before we close, let's push it just a little bit farther.

```php
<?php
	include_once "../../secure/connection.php";

	$Scarlet = new iSeries;

	$connS = $Scarlet->connect();
	if (!$connS) {
		echo "<br />\n no connection!!!";
		}
	else {
		echo "<br />\n Scarlet Connection OK ";
		}

	$Begonias = new iSeries;
	$connB = $Begonias->connect();

	if (!$connB) {
		echo "<br />\n  no connection!!!";
		}
	else {
		echo "<br />\n Begonias Connection OK ";
		}

	$rcS = $Scarlet->close();
	if ($rcS) {
		echo "<br />\n  Scarlet Connection was successfully closed.";
		}
	else {
		print "<br />\n nope...not yet.";
		}

	$rcB = $Begonias->close();
	if ($rcB) {
		echo "<br />\n Begonias Connection was successfully closed.";
		}
	else {
		print "<br />\n nope...not yet.";
		}
?>
```



>Scarlet Connection OK
>Begonias Connection OK
> Scarlet Connection was successfully closed.
> Begonias Connection was successfully closed.

Credit where credit is due:

Obviously, the folks doing all this PHP work and the Zend folks...they've made it very, very easy for us to start exploiting this fun language.

Bryan Jamison, the “young kid” in my own shop that continues to humor me and assist with “PHP for Dinosaurs” (his term, not mine)... ...everyone needs a smirking 20something in his shop.

Bob Dusek, my all time favorite percussion player that just happens to be a pick-a-language expert who also humors me and returns every wtf email with lengthy examples and explanations (everyone needs a friend like this!)