

Persistent Static Variables Across Instances in PHP

Thursday, June 12th 2008

Wait, What? Yeah that's what I thought too. Still no Idea what I'm talking about? Well, let's take a look at the following code. Let's call it "fred".

```
class foo {
    function bar( $b = 0 )
    {
        static $a = 0;
        if ( $b ) {
            $a = $b;
        }
        echo $a;
    }
}
```

```
$faz = new foo;
$faz->bar(3);
$baz = new foo;
$baz->bar();
foo::bar();
foo:bar(1);
$faz->bar();
```

The code above, named "fred", basically creates a static variable \$a inside the function foo(). When you call foo(); it outputs the value of \$a. When you call foo('x');, where x can be anything, it updates the value of \$a with 'x', and outputs the new result.

Now, what would expect "fred" to output? If your like me, then you are completely wrong. "fred" will actually output the following code.

```
/*
Actual Outputs:
$faz->bar(3); ==> 3
$baz->bar(); ==> 3
foo::bar(); ==> 3
foo:bar(1); ==> 1
$faz->bar(); ==> 1

Expected outputs:
$faz->bar(3); ==> 3
$baz->bar(); ==> 0
foo::bar(); ==> 0
```

```
foo:bar(1);      ==> 1
$faz->bar();    ==> 3
*/
```

Yes, that's what I said at the start, "Persistent Static Variables Across Instances". The static variable \$a actually persists across the two instances of foo that "fred" created, and even into the static method call. This was completely unexpected, at least by me. So I'll ask, does anyone know if this is actually the expected behaviour, and why it is or is not?