

Re: Classes – Dumb question

Source: <http://coding.derkeiler.com/Archive/PHP/php.general/2007-10/msg00496.html>

- *From:* ZeldorBlat <zeldorblat@xxxxxxxx>
 - *Date:* Thu, 11 Oct 2007 08:41:09 -0700
-

On Oct 11, 8:57 am, japr...@xxxxxxxxxxx (Jason Pruim) wrote:

On Oct 11, 2007, at 8:36 AM, Jay Blanchard wrote:

[snip]
okay, this is really (!) embarrassing, but I have to ask:

Why would I want to use classes in PHP?

I have been using PHP for years now and writing the "normal" functions all the time. I have never even bothered working with classes, but now I would love to know what makes the classes so special...

Please go easy on me ;o) Just trying to make another step :o)
[/snip]

Do not be embarrassed, this is a very good question.

First of all what you call "normal" is procedural or functional programming. There is nothing wrong with doing things this way and may be especially quick and efficient when doing basic web sites and applications. Document well and you will have no problem maintaining your code.

Re: Classes – Dumb question

OOP (object oriented programming) is especially useful when the application you have created needs to scale. A quick example; you have sold your products to the consumer market for a long time but now the commercial market has become interested. Commercial customers are different than non-commercial customers, different data, different credit requirements, different shipping, etc. but they still have a lot in common, If you had a class Customer you could extended that class to include commercial customers and only have to code for the unique qualities of that kind of customer. Then if another type of customer crops up, say a military contract, you could extend again;

```
class Customer {  
....  
}
```

```
class CommercialCustomer extends Customer {  
/*  
*only code unique to commercial customers  
* inherits from Customer other variables  
* and functions that are common  
*/  
}
```

```
class MilitaryCustomer extends Customer {  
/*  
*only code unique to military customers  
* inherits from Customer other variables  
* and functions that are common  
*/  
}
```

<http://www.sitepoint.com/article/object-oriented-php>

—
PHP General Mailing List (<http://www.php.net/>)
To unsubscribe, visit:<http://www.php.net/unsub.php>

Re: Classes – Dumb question

Not trying to hijack the thread... Hopefully this is related enough, if not I apologize. Would a good use of a class be to write a generic database connection script? and then feed in the different variables, such as customer login, database, stuff like that?

```
something like class DBConnect {  
  // Connect to database  
  mysql_connect($server, $login, $password, $database);  
}
```

or no?

--

Kinda sorta. The idea with classes is to expose an interface rather than an implementation. That enables you to change things inside the class without having to change how you use the class.

Consider the following interface:

```
interface Database {  
  public function __construct($server, $username, $password,  
    $database);  
  public function query($sql);  
}
```

Suppose you're using a MySQL database. Then you might have a MySQL class that looks something like this (this is a simplified example):

```
class MySQL implements Database {  
  private $conn;  
  
  public function __construct($server, $username, $password,  
    $database) {  
    $this->conn = mysql_connect($server, $username, $password,  
      $database);  
  }  
  
  public function query($sql) {  
    return mysql_query($sql, $this->conn);  
  }  
}
```

So, in your application, you would use this class like this:

```
$db = new MySQL('foo', 'bar', 'baz', 'xyzy');  
$sql = 'select * from sometable';  
$rs = $db->query($sql);
```

Re: Classes – Dumb question

Now let's say you change over to Postgres for your database. Now you create a class like this:

```
class Postgres implements Database {
private $conn;

public function __construct($server, $username, $password,
$databse) {
$conn_string = "host=$server port=5432 dbname=$databse user=
$username password=$password";
$this->conn = pg_connect($conn_string);
}

public function query($sql) {
return pg_query($this->conn, $sql);
}
}
```

Now your code looks like this:

```
$db = new Postgres('foo', 'bar', 'baz', 'xyzyzy');
$sql = 'select * from sometable';
$rs = $db->query($sql);
```

Notice how the details of connecting and running a query (like the functions used, for instance) are encapsulated inside the class. When I go to actually use the database all I need to know is that there is some method called "query" which, given some SQL, runs the query and returns a result. I know that both the MySQL class and the Postgres class have a method called query() because they both implement the "Database" interface. I don't care about the specifics of what happens inside the class — but I know I can treat both objects as though they are the same (that's called "polymorphism") because they are both "Databases."

Does that make sense?

.