OOP Objektorientierte Programmierung in PHP - Part 1

Hallo liebe Community!

Dies ist mein erstes Tutorial also seit nicht zu streng mit der Kritik, über Verbesserungsvorschläge würde ich mich dennoch freuen!

Ich setze voraus, dass man weiß wie Funktionen geschrieben werden und dass man mit Variablen umgehen kann.

Im ersten Part des Tutorials möchte ich erstmal die basics sowie die Funktionen von Klassen erklären.
Nehmen wir mal an wir programmieren ein Browsergame mit Raumschiffen. Es gibt Zerstörer, Jäger und Transporter. Jetzt sollten ja alle Zerstörer ungefähr gleich sein; alle Jäger ungefähr gleich; alle Transporter ungefähr gleich. Zum Beispiel muss jeder Zerstörer mit Weltraumlasern scheißen können, wie gut die jedoch sind hängt von der Forschungsstufe des Spielers ab. Jedoch haben alle Raumschiffe auch etwas gemeinsam: sie können fliegen und haben eine bestimmte Anzahl an Leben!
Sehen wir uns das jetzt erstaml als code an:

Quellcode

```
<?php
class Raumschiff //Klasse mit dem Namen "Raumschiff"
{
public $leben = 100; //Die Eigenschaft "$leben". Sie ist "public" (öffentlich) zugänglich. Variablen (und auch Konstante) die
//zu Klassen gehören nennt man "properties" (Eigenschaften).
public function fliegen() //Die Methode "fliegen". Funktionen die zu Klassen gehören nennt man "methods" (Methoden).
//Die Methode ist public.
{
echo "Und wir fliegen weiter!";
}
}
# To be continued
```

Alles anzeigen

Der Code oben tut so im Grunde nichts. Er gibt nur an was die Klasse Raumschiff alles beinhaltet. In diesem Fall das Datenfeld (Eigenschaft, property) "$leben" (100) und die Methode (method) "fliegen()". Properties werden gewöhnlich (alleine schon wegen der Übersichtlichkeit) vor den metods deklariert. Sie müssen aber noch keinen Wert bekommen (kein Initialisierungs Zwang).

Um nun mit der Klasse zu arbeiten müssen wir sie instanzieren (eine Instaz dieser Klasse erzeugen) oder simpel: ein Objekt der Klasse erstellen:

Quellcode

```
$UnserRaumschiff = new Raumschiff(); //Variablen, die ein Objekt einer Klasse enthalten, werden meistens am Anfang groß geschrieben
//Ein Objekt einer Klasse erzeugt man mit dem statement new und dem Klassen-
//namen gefolgt von runden Klammern "()". Es ähnelt einem Funktionsaufruf, bis auf das wir
//ein neues (new) object erhalten.
$UnserRaumschiff->fliegen(); //Man kann nun über das object auf alle public methods zugreifen. Dies geschieht mit einem
//Zugriffsoperator. Meistens ist dies ein Pfeil "->". Später werden wir einen weiteren kennen lernen
echo $UnserRaumschiff->leben; //Genau wie man auf methods zugreifen kann, so kann man auch auf public properties zugreifen
?>
```

Alles anzeigen

Über den "->" Pfeil kann man auf die Eigenschaften und Methoden von Objekten zugreifen. Dies gilt nur für Methoden und Eigenschaften die öffentlich sind.
Es gibt auch private Methoden und private Eigenschaften. Auf diese kann nur das Objekt selbst zugreifen. Jedoch stellt sich schnell die Frage: "Was sollen einem die ganzen privaten (private) Methoden und privaten Eigenschaften bringen?" Nunja, bei 20 Zeilen Code -> garnichts! Aber ab 50 Zeilen wird das Programm übersichtlicher wenn man auf die Eigenschaften von den Objekten nur über sogenannte "get"-Methoden (gib) und "set"-Methoden (setze) zugreift (Die Deutschen Begriffe habe ich bis auf in unserem SEHR deutschen Informatik Unterricht NIE gebraucht! Das gilt eigentlich für alle deutschen Begriffe, jedoch sollte man die meisten kennen.) Die get & set Methoden werden umgangssprachlich auch "getter" und "setter" genannt, jedenfallls so ähnlich ;).

> OK Übersichtlichkeit ist ein Punkt, aber letztendlich geht es bei OOP um Datenkapselung, was bedeutet, dass nicht jeder von außen Eigenschaften eines Objektes ändern soll , wie er lustig ist. Erst mit gettern und settern erhält das Objekt die volle Kontrolle über seine Eigenschaften und deren Zuweisungen
> *Gambler*

Danke an Gambler für seine PN ;)! Mit dem oben möchte er sagen, dass wirklich nur das Objekt selbst die Daten speichert und so auch keine "falschen" Daten zugewiesen werden können!
Gucken wir uns unser Raumschiff mal mit private properties an:

Quellcode

```
1. <?php
2. class Raumschiff
3. {
4. private $leben = 100; //Die Leben in privater Form
5. public function fliegen()
6. {
7. echo "Auf in den Hyperraum";
8. }
9. public function getLeben()
10. {
11. return $this->leben; //Was $this ist wird unten erklärt
12. }
13. public function setLeben($leben) //Methoden dürfen auch Parameter erhalten
14. {
15. if(is_numeric($leben) && $leben >= 0) //Dank der setFunction kann die Klasse Daten auf Gültigkeit überprüfen
16. {
17. $this->leben = $leben;
18. }
19. }
20. }
21. $Schiff = new Raumschiff();
22. $Schiff->fliegen(); //Fliegen können wir noch ganz normal!
23. //echo $Schiff->leben; //Würde einen Error erzeugen, da versucht wird, auf etwas zuzugreifen, obowhl es kein
24. //Zugriffsrecht gibt
25. echo $Schiff->getLeben();
26. ?>
```

Alles anzeigen

Die erste Frage die sich beim Programmieren einer getFunction stellt ist, wie soll man auf die Eigenschaft des Objekts zugreifen, dessen Methode aufgerufen wurde? Dafür wurde das Schlüsselwot $this eingeführt. Es "enthält" das "aktuelle" Objekt (eigentlich referenziert es das eigene Objekt, aber Gambler will dazu keine Begriffserklärung schreiben), da man von einer Klasse auch mehrere Objekte erzeugen kann (es gibt ja nicht nur ein Raumschiff). Außerhalb der Klasse kann man nicht auf eine Methode oder eine Eigenschaft die private ist, zugreifen. Innerhalb der Klasse, also in den Methoden der Klasse, ist dies jedoch möglich. So lassen sich ganz einfach get&setFunctions schreiben.

Das sind eigentlich schon die Basics. Für mich jedenfalls. Verbesserungsvorschläge & Anmerkungen gerne per PM. In Part 2 werde ich auf Konstruktoren und Destruktoren eingehen: [wiki]OOP Objektorientierte Programmierung in PHP - Part 2

[/wiki]

n0x-f0x