PHP
Aufbaukurs

Tag 3. PHP5 & Klassen

Igor Olkhovskiy

Dr. Dipl.- Ing.

Kontakt: olkhovskiy@rrzn.uni-hannover.de

- **PHP/FI (“PHP 1” und “PHP 2”):** Keine objektorientierten Sprachmerkmale
- **PHP 3**: Erster Versuch, objektorientierte Programmierung in PHP zu ermöglichen
- **PHP 4**: Zend Engine 1: Verbesserte Unterstützung für objektorientierte Programmierung, aber stark limitiertes Objektmodell
- **PHP 5**: Zend Engine 2: Java-ähnliches, von Grund auf überarbeitetes Objektmodell

- **Modularisierung**: Zerlegung des Software-Systems in autonome Einheiten
- **Wiederverwendung**: Bereits implementierte Einheiten sollen zu einem neuen Ganzen zusammengesetzt werden können
- **Erweiterbarkeit**: Bestehende Einheiten sollen um neue Funktionalitäten erweitert werden können
- **Abstraktion und Kapselung**: Der Verwender einer solchen Einheit nutzt deren öffentliche Methoden und Variablen
- Schnittstelle, Implementierungsdetails werden versteckt.

- **Klasse**: Bauplan für die Erzeugung von **Objekten**.
- Klasse kapselt Daten (**Attribute**) und Operationen (**Methoden**) in einer Einheit.
- Jedem Objekt lässt sich eine Klasse zuordnen.
- Die Objekten können die Methoden und Variablen gemeinsam nutzen.
- Die Objekte einer Klasse unterscheiden sich nur in ihrem Zustand von einander.

```
<?php

class MyClass
      {
            var $str1=array("H","i","!");
            var $z      ahl1 = 2004;
      }
//Definierung
 $obj1 = new MyClass();

//Ausgabe
 echo $obj1->zahl1."<br />“;
 echo $obj1->str1[0].$obj1->str1[1].$obj1->str1[2]."<br />";

?>
```

```
<?php
    class MyClass
            {
            var $str1=array("H","i","!");
            var $zahl1 = 2004;
                    function fn1()
                    {
                    $a = 2005;
                    echo $a;
                     }
             }

    $obj1 = new MyClass();
    echo $obj1->zahl1."<br />";
    echo $obj1->$str1[0].$obj1->str1[1].$obj1->str1[2]."<br />";
    echo $obj1->fn1();
?>
```

```
<?php
class Bankkonto
    {
        function __construct()
        {
        print "Neues Konto erzeugt<br>";
        }
        function __destruct()
        {
        print "Ueberweisung Restbetrag auf Direktorkonto<br>";
        }
    }

$bk = new Bankkonto();

?>
```

- Trennung von Nutzungs- und Implementierungsschicht, von Realisierung und Nutzung
- Zu diesem Zweck kann den Attributen und Methoden einer Klasse in PHP 5 bei ihrer Deklaration eine von drei möglichen Sichtbarkeiten zugewiesen werden:
- **private** kennzeichnet die lokale Sichtbarkeit
  - Das Element ist nur in der umgebenden Klasse sichtbar
- **protected** kennzeichnet die eingeschränkte Sichtbarkeit
  - Das Element ist nur in der umgebenden Klasse und deren Kindklassen sichtbar
- **public** kennzeichnet die öffentliche Sichtbarkeit
  - Auf das Element kann von Objekten sämtlicher Klassen zugegriffen werden
  - Wird keines der drei Schlüsselwörter bei der Deklaration angegeben, so wird implizit public angenommen

In PHP 4 gab es nur die öffentliche Sichtbarkeit (var)

Die Vererbungsbeziehung einer Kindklasse zu ihrer Elternklasse wird durch das extends-Schlüsselwort in der Klassendeklaration ausgedrückt.

Mit dem Schlüsselwort **final** markierte Methoden der Elternklasse können in der Kindklasse nicht redefiniert werden, als final markierte Klassen können per Vererbung nicht erweitert werden.

```
<?php
class BasisClass {
      var $str1=array("H","i","!");
      var $zahl1 = 2004;
function BasisClass()
{
      echo "Constructor.BasisClass:";<br />";
      echo $this->str1[0].$this->str1[1].$this->str1[2]."<br />";
}
 function fn1b() { $a = 2005; echo "fn1:Locale Variable a = ".$a"; }
 function fn2b($b="1005")  { $a = 2005; echo "fn2-Parameter = ".$b".
      "<br />"; echo "fn2:Globale Variable zahl1 = ".$this->zahl1."<br />"; }
}
```

```
//Abgeleitete Class class
KindClass extends BasisClass
{
 function KindClass($ueberschrift)
                { echo $ueberschrift; }
}
```

```
<?php
//Public, protected und private

class foo {
  public function public_foo() {  print("I'm public<br/>");   }
  protected function protected_foo() {   $this->private_foo(); print("I'm protected<br/>");  }
  private function private_foo() { $this->x = 3; print("I'm private<br/>"); }
}
class foo2 extends foo {
        public function display() {
          $this->protected_foo();
          $this->public_foo();
    // $this->private_foo();  // Invalid! the function is private in the base class
}
}
```

```
$x = new foo();
$x->public_foo();

$x2 = new foo2();
$x2->display();
?>
```

- Abstrakte Klassen werden nicht instanziert, sie dienen nur dazu, gemeinsame Methoden

  und Eigenschaften zu definieren, die später von einer Anzahl ähnlicher Erben verwendet werden.

- Abstrakte Klassen sind deswegen so schön, weil sie die Dinge, die die Erben unterscheiden, gar nicht implementieren, sondern mehr oder weniger sagen können: Diese Methode muss jeder einzelne meiner Erben für sich implementieren.

```
<? php

  abstract class cAbstractAnimal {

  function getNumberFeets()   {  return 4;  }
  abstract function getCommunicationMethod();

}

class cHuman extends cAbstractAnimal {

  function getNumberFeets()   {   return 2;   }
  function getCommunicationMethod()   { return „bellen";   }
}
echo „Bären haben ".$human->getNumberFeets()." Füße
            und ".$human->getCommunicationMethod()."<br>";
?>
```

Quelle : www.sebastian-bergmann.de (Sebastian Bergmann, Kurs: Vertiefende PHP Schulung).