Auf einen Blick

Inhalt

Vorwort

Für Uli Kruppe, der uns zu früh verlassen hat

PHP geht in die fünfte Runde und wird endlich erwachsen. Die mit 15205474 Installationen populärste Sprache im Internet[1] wandelte sich im Laufe der Jahre von einer Hand voll Hilfsfunktionen für Web-Master zur professionellen Entwicklungsplattform.

Das Einsatzspektrum von PHP reicht vom simplen Gästebuch-Skript bis hin zu komplexen Online-Shops oder Content-Management-Systemen. Seine Beliebtheit verdankt PHP nicht zuletzt der engen Anbindung an Datenbanken wie MySQL oder PostgreSQL.

Dieses Buch

PHP 5 – Die Neuerungen gibt Ihnen einen schnellen Überblick über die neuen Fähigkeiten von PHP 5: die neue Engine, SQL-Lite, neue XML-Funktionen und vor allem über die objektorientierte Programmierung.

Das Buch setzt voraus, dass Sie sich bereits mit PHP 4 auskennen. Kurze, knackige Beispiele zeigen im Detail, was sich geändert hat. An gegebener Stelle erfahren Sie, wie und wofür sich die neuen Techniken eignen und wie Sie diese umsetzen. Das Ziel: schnelle Information für Könner.

Im ersten Kapitel finden Ungeduldige ganz schnell Informationen. Die wichtigsten Neuerungen werden angesprochen und mit kurzen Beispielen ausgeführt. Die nachfolgenden Kapitel behandeln ausführlich die neue Technik und vertiefen sie mit weiteren, ausführlicheren Programmbeispielen.

Da die Entwicklung von PHP 5 mehr Zeit brauchte als erwartet, stützen sich die Beispiele in diesem Buch auf die Beta 4 von PHP 5. Das ist die letzte Beta-Version; sie entspricht im Funktionsumfang der endgültigen Version.

Kurze Geschichte von PHP

PHP begann als »Personal Homepage Tools«. Das waren ein paar, ursprünglich in Perl, später in C programmierte Funktionen, um Zugriffe auf einen Webserver zu zählen. Der Autor, Rasmus Lerdorf, »spielte« außerdem mit ein paar Funktionen, um SQL-Abfragen vom Webserver aus auszuführen.

1 laut Netcraft, März 2004

Diese beiden Spielereien mündeten 1995 in PHP/FI. Dieses Kürzel stand für »Personal Home Page / Forms Interpreter« und wurde 1997, komplett neu geschrieben, als PHP/FI 2.0 veröffentlicht.

Der Durchbruch kam 1997, als Andi Gutmans und Zeev Suraski PHP wieder einmal komplett neu schrieben, um es für ein eigenes Projekt zu verwenden. In Zusammenarbeit mit Rasmus Lerdorf entstand so PHP 3. Dank der offenen Spracharchitektur und der leichten Erlernbarkeit brachte es PHP 3 schnell weiter. Provider installierten die Sprache auf ihren Servern, Web-Master freuten und freuen sich über die einfache Einbindung der Sprache in HTML und die eingebaute Schnittstelle zur Datenbank MySQL. Der Erfolg dieser Datenbank ist eng mit dem Erfolg von PHP verknüpft: Das Gespann ist die Grundlage unzähliger Content-Management-Systeme und E-Commerce-Anwendungen.

Mit PHP 4 wuchs die Sprache im Jahr 2000 weiter: Verbesserte Modularität, Session-Management oder die Pufferung von Ausgaben gehörten zum Funktionsangebot. Ein neu entwickelter Kern trug dem Bedarf Rechnung, auch komplexe Anwendungen effizient umzusetzen.

PHP 5 macht einen konsequenten Schritt in Richtung objektorientierter Programmierung. PHP ist endlich erwachsen geworden. Was neu ist, lesen Sie in diesem Buch.

Die Autoren

Markus Schraudolph, 37, ist Journalist und Programmierer. Er schreibt seit 16 Jahren Bücher und Fachzeitschriften-Artikel zu allen Themen aus dem PC-Umfeld. Seine Schwerpunktgebiete als Programmierer sind die Web-Programmierung und Datenbanken.

Martin Goldmann, Jahrgang 1965, arbeitet seit 1986 als Journalist für IT-Fachzeitschriften wie *Internet Professionell*, *PC-Direkt*, *Chip* oder *Computerbild*. Seine Themenschwerpunkte sind Netzwerke und die Programmierung von Internet-Auftritten.

Gemeinsam haben die Autoren das Content-Management-System *Tippscout.de* programmiert und betreuen das gleichnamige Internet-Portal.

Die Autoren bedanken sich bei ihren Frauen Gaby und Franziska für viel Geduld und Zuwendung.

Dank geht auch an Claus Herwig, den geduldigen Provider, und an Reinhard Weber für seine technische Unterstützung.

Martin Goldmann und Markus Schraudolph
April 2004

1 Die wichtigsten Neuerungen auf einen Blick

Dieses Kapitel zeigt Ihnen die Neuerungen von PHP 5 im Schnelldurchlauf. Verschaffen Sie sich hier einen ersten Überblick, und probieren Sie die kurzen Beispiele aus.

1.1 PHP 5 ist objektorientiert

Bislang war PHP nur halbherzig objektorientiert. Zwar konnten Programmierer Klassen schaffen und ihnen Methoden geben. Doch bei OOP-Standards wie dem Überladen von Klassen oder Zugriffsmodifizierern wie »private« und »protected«, die Instanzvariablen vor der Außenwelt abschirmen, musste das alte PHP passen. Einheitlich benennbare Konstruktoren und Destruktoren fehlten ebenso wie die Möglichkeit, Default-Aktionen in Klassen zu definieren.

Mit PHP 5 steigt die Skript-Sprache in die Liga der objektorientierten Sprachen auf. Wer mit den Konzepten von Java, C++ oder C# vertraut ist, wird sich mit PHP schnell zurechtfinden.

Einsteiger und Freunde der strukturellen Programmierung müssen dennoch nicht verzagen. PHP bleibt selbstverständlich rückwärtskompatibel. Sie können also genauso schnell und effektiv Skripten schreiben wie zuvor. Und ein paar neue Funktionen liefert PHP 5 gleich mit dazu.

1.1.1 Beispiel: Private und Protected-Variablen und -Methoden

Objektorientierte Puristen haben über PHP 4 immer die Nase gerümpft – zu Recht: Denn in der Vierer-Version war es nicht möglich, Variablen und Methoden innerhalb einer Klasse vor einem Zugriff von außen zu schützen.

Gerade Programmierer, die in gemeinsamen Projekten arbeiten oder ihren Code weitergeben, brauchen aber ein solches Feature. Denn wenn jede Variable und Methode von außen zugänglich ist, schafft das zusätzliche Fehlerquellen: Jeder andere Programmierer kann jederzeit eine beliebige Variable im Objekt ändern und damit unerwünschte und unerwartete Resultate erzeugen.

Aber in PHP 5 ist das jetzt anders. Sie als Programmierer einer Klasse bestimmen, ob eine Variable von außen zugänglich ist oder nicht. Folgende Möglichkeiten gibt es:

- Die Variable ist von außen zugänglich, darf gelesen oder geändert werden. Eine solche Variable deklarieren Sie als `public`.[1]
- Die Variable ist nur innerhalb der eigenen Klasse und in den Klassen zugänglich, die abgeleitet werden. Sprich: Eine Klasse, die eine andere beerbt, darf diese Variable lesen oder auch ändern. Das hierfür zuständige Schlüsselwort ist `protected`.
- Kompletten Schutz bietet das Schlüsselwort `private`. Eine so deklarierte Variable darf weder von außen gelesen werden noch ist sie einer vererbten Klasse zugänglich.

Am folgenden Beispiel sehen Sie die Wirkung der drei unterschiedlichen Zugriffsmodifizierer auf die Sichtbarkeit der Instanzvariablen:

```
<?php
/**
* Beispiel fuer public, private und protected
*
*/
   class TestClass
   {
      public $strPub = "Ich bin public";
      protected $strProt = "Ich bin protected";
      private $strPriv = "Ich bin private";

      function showVars()
      {
         echo "Das sieht die Klasse <b>TestClass</b>:<br>\n";
         echo "\$strPub: $this->strPub<br>\n";[2]
         echo "\$strProt: $this->strProt<br>\n";
         echo "\$strPriv: $this->strPriv<br>\n";
         echo "<br><br>\n";
      }
   }

   class ExtClass extends TestClass
   {
```

1 Damit Ihre in PHP 4 geschriebenen Klassen weiterhin funktionieren, ist Public die Standardeinstellung für Instanzvariablen. Es ist also egal, ob Sie einer Variablen das Schlüsselwort `var` oder `public` voranstellen, in beiden Fällen kann von außen auf sie zugegriffen werden.

2 Wundern Sie sich über den umgedrehten Schrägstrich vor dem Dollarzeichen? Hier möchten wir in der Ausgabe den Namen der Variablen haben. Ohne den Schrägstrich würde der automatische Ersetzungsmechanismus von PHP den Wert der Variablen hier einfügen wollen. Der Schrägstrich verhindert also, dass das Dollarzeichen als Variablen-Präfix interpretiert wird.

```
      function showVars()
      {
         parent::showVars();
         echo "Das sieht die Klasse <b>ExtClass</b>:<br>\n";
         echo "\$strPub: $this->strPub<br>\n";
         echo "\$strProt: $this->strProt<br>\n";
         // Die folgende Zeile zeigt nichts an
         echo "\$strPriv: $this->strPriv<br>\n";
         echo "<br><br>\n";
      }
   }

   $objTest = new TestClass();
   $objExt = new ExtClass();

   // Die Variablen innerhalb der Klasse ExtClass anzeigen
   $objExt->showVars();

   // Variablen ansehen
   echo "Das sieht das Hauptprogramm:<br>\n";
   echo "\$objTest->strPub: $objTest->strPub<br>\n";
   //echo "\$objTest->strProt: $objTest->strProt<br>\n";
   // echo "\$objTest->strPriv: $objTest->strPriv<br>\n";
   echo "<br><br>\n";

   // Aenderungsversuche
   // Eine public-Variable aendern
   $objTest->strPub =
   "Ich bin public, vom Hauptprogramm geändert!";
   // Eine protected-Variable aendern --> FEHLER
   // $objTest->strProt =
   // "Ich bin protected. Das Hauptprogramm kann mich nicht
   //  ändern";
   // Eine private Variable aendern --> FEHLER
   // $objTest->strPriv =
   // "Ich bin private. Das Hauptprogramm kann mich
   //  nicht ändern";

   $objTest->showVars();
?>
```

Listing 1.1 Die Wirkung der neuen Anweisungen `public`, `private` und `protected` auf die Sichtbarkeit von Instanzvariablen.

In der `TestClass` deklarieren Sie drei Variablen (eine als `public`, eine als `protected`, eine als `private`) und belegen sie mit je einem String. Dazu liefert unsere Testklasse noch eine Methode `showVars()`, die einfach nur den Inhalt der Variablen anzeigen soll. Danach bauen Sie noch eine zweite Klasse auf: `ExtClass` erweitert unsere `TestClass`. Diese Klasse hat nicht mehr zu bieten als eine weitere `showVars()`-Methode. Die `ExtClass` ruft erst einmal dieselbe Methode bei der übergeordneten Klasse auf und gibt danach selbst den Inhalt ihrer Variablen aus.

Und genau an dieser Stelle stoßen wir auf den ersten Fehler, den dieses Skript erzeugt: Die dritte Zeile

```
echo "\$strPriv: $this->strPriv<br>\n";
```

gibt nichts aus. Warum? Weil `ExtClass` als Nachkomme keinen Zugriff auf die als `private` deklarierte Variable `$strPriv` aus der Elternklasse hat. Je nach PHP-Installation erscheint übrigens eine Warnung wie:

```
Notice: Undefined property: extclass::$strPriv
```

Das Hauptprogramm instanziiert je einmal beide Klassen und ruft dann die Anzeige der Variablen auf.

Danach zeigt es den Inhalt der Variablen `$strPub` an. Unter der entsprechenden Zeile finden Sie noch zwei weitere, auskommentierte Zeilen. Kommentieren Sie eine von beiden ein, und starten Sie das Programm erneut – der Fehler ist fatal:

```
Fatal error: Cannot access protected property testclass::$strProt
```

In den letzten Zeilen versucht das Hauptprogramm, eine Variable zu ändern. Mit der als `public` definierten Variablen `$strPub` funktioniert das auch. Wenn Sie aber eine der folgenden Zeilen auskommentieren, gibt es wieder einen fatalen Fehler.

Genauso wie Variablen lassen sich Methoden schützen. Zum Beispiel:

```
<?php
/**
* Beispiel fuer private Methoden
*
*/
   class TestClass
   {
      private function testMethod()
```

```
      {
         echo "Ich bin eine private Methode";
      }

      public function showText()
      {
         $this->testMethod();
      }
   }

   $objTest = new TestClass();

   // Funktioniert
   $objTest->showText();

   // FEHLER!
   $objTest->testMethod();
?>
```

Listing 1.2 Genauso wie Instanzvariablen können Sie bei den Methoden entscheiden, ob sie auch von außerhalb der Klasse erreichbar sein sollen oder nicht.

Die Klasse `TestClass` hat zwei Methoden. Die eine, als `private` deklarierte `testMethod()`, gibt einen kurzen Text aus. Die andere, `showText()`, ruft wiederum `testMethod()` auf.

Im Hauptprogramm wird zunächst `showText()` aufgerufen. Als Resultat auf dem Bildschirm sehen Sie den Text `Ich bin eine private Methode`. Danach probiert das Hauptprogramm, direkt `testMethod()` aufzurufen – und scheitert mit:

```
Fatal error: Call to private method testclass::testMethod()
```

So einfach ist es, Methoden zu schützen. Die Abstufungen gelten wie bei den Variablen.

Der Vorteil von `private` und `protected`: Der gesamte Programmcode wird robuster und leichter zu handhaben, wenn Variablen in Objekten vor unabsichtlichen Änderungen geschützt werden. Der kleine Nachteil: Der Aufwand erhöht sich ein wenig: Als Programmierer sollten Sie sich immer genau überlegen, welche Variablen und Methoden einer Klasse Sie exponieren.

Und was passiert mit den alten Variablen und Methoden in Ihren Klassen aus PHP 4? Sie werden weiterhin `public` zugänglich sein. Sofern Sie also kein anderes Schlüsselwort eingeben, bleibt alles beim Alten. Aber es lohnt sich, viel benutzte Klassen zu durchforsten und mit `private` und `protected` abzusichern.

1.1.2 Abstrakte Klassen

Abstrakte Klassen erlauben es, »Modelle« für Klassen herzustellen. Anhand dieser Modelle kann ein Programmierer dann seine eigenen Klassen bauen. Der Vorteil des Ganzen: Ein Programmierer kann mit solchen abstrakten Klassen unveränderliche Klassen bauen, auf denen viele weitere Klassen basieren dürfen. Da sie nicht verändert werden können, haben andere Programmierer die Gewissheit, dass der Code robust und flexibel wiederverwendbar ist.

1.1.3 Interfaces

Interfaces haben in PHP 5 eher eine Sicherungsfunktion. In einem Interface, zu Deutsch: einer Schnittstelle, legen Sie ein Set von Methoden fest. Wird dieses Interface in eine Klasse implementiert, so müssen für die Klasse Funktionen mit den im Interface festgelegten Namen programmiert werden. Fehlt eine Methode, meldet der Compiler einen Fehler.

Wozu dient das Ganze? Es erleichtert die Kontrolle über den Code und die Programmierung, sobald mehr als ein Programmierer an einem Projekt arbeitet. Über ein Interface legt man bestimmte Richtlinien fest, die jede Klasse einhalten muss – so läuft man weniger Gefahr, etwas zu vergessen, und der Code wird robuster.

1.1.4 Beispiel: Fehlerbehandlung mit try...catch

Mit `try` und `catch` bietet PHP endlich ein einheitliches Verfahren, um Fehler abzufangen und zu verarbeiten. Das Prinzip: Ein Codeblock wird von `try{}` umschlossen. Tritt innerhalb dieses Blocks ein Fehler auf, so fängt ihn `catch{}` ab.

Für das Erzeugen der Fehlermeldung ist `throw` verantwortlich. Sobald etwas schief geht, erzeugt diese Anweisung ein neues Objekt der Klasse `Exception` und übergibt ihr den Text einer Fehlermeldung. Selbstverständlich können Sie auch eigene Klassen zur Fehlerbehandlung erzeugen, um beispielsweise nach einem Fehler notwendige Aufräumarbeiten zu veranlassen oder genauere Meldungen auszugeben.

Das folgende Beispiel zeigt das Prinzip von `try` und `catch`:

```
<?php
/*
* Eine Beispielklasse fuer die Fehlerbehandlung
*/
class Calculation
{
// Diese Funktion teilt $x durch $y
function divide($x,$y)
{
  // Pruefen, ob $y gleich 0, falls ja -> Fehlermeldung erzeugen
  if($y == 0)
    throw new Exception("Divisionsfehler (\$y = $y)");
  // Ergebnis zurueckgeben
  return $x/$y;
  }
}

// Hauptprogramm
$objCalc = new Calculation();

try
{
  // Absichtlich einen Fehler erzeugen
  $intResult = $objCalc->divide(99,0);
  // Diese Meldung erscheint nur, wenn kein Fehler auftritt
  echo 'Das Ergebnis: '.$intResult.'<br />';
}
// Die Fehlermeldung auffangen
catch (Exception $e)
{
  // Methode zum Ausgeben der Meldung
  echo $e->getMessage();
}
?>
```

Listing 1.3 Mit den neuen Befehlen `try` und `catch` haben Sie bessere Möglichkeiten, auf Fehler in Ihren Skripten zu reagieren.

Der große Vorteil: Beim Programmieren von Klassen und Funktionen müssen Sie sich nur noch darum kümmern, ob und mit welcher Begründung abgebrochen und eine Fehlermeldung erzeugt wird. Wie das Programm schließlich mit dem Fehler umgeht, regeln Sie später im Hauptprogramm.

1.1.5 Beispiel: Erzeugen von Properties

Mit den neuen Schlüsselwörtern `protected` und `private` schützen Sie Instanzvariablen vor dem direkten Zugriff durch Skripten. In gewisser Weise setzen die neuen Funktionen `__get` und `__set` noch eins drauf. Denn wenn man sie in einer Klasse implementiert, werden sie immer dann aufgerufen, wenn der Zugriff auf eine undefinierte Instanzvariable erfolgt.

Anhand des per Parameter übergebenen Namens dieser unbekannten Instanzvariablen lässt sich dann entscheiden, was zu tun ist. So könnte beispielsweise eine Klasse, die Wetterdaten eines Ortes enthält, eine Instanzvariable für die Temperatur in Grad Fahrenheit anbieten, obwohl intern die Temperatur nur in Grad Celsius gespeichert wird:

```
<?php
// Beispiel
// Die Verwendung von Property-Methoden
class Wetter {

  // Instanzvariablen
  var $nTempC;
  var $nLuftdruckHPa;
  var $nNiederschlag;

  // Property-Methoden
  function __get($strVarname) {
    if ($strVarname=='nTempF'){
      // Umrechnung Celsius -> Fahrenheit
      return (($this->nTempC * 9 / 5) + 32 );
      }
    else  {
      return ("Variable '$strVarname' unbekannt!");
      }
    }

  function __set($strVarname,$xValue) {
    if ($strVarname=='nTempF'){
```

```
      // Umrechnung Fahrenheit -> Celsius
      $this->nTempC = ($xValue- 32) * 5 / 9;
      $this->nTempC = round($this->nTempC*10) / 10;
       }
    else  {
      echo("Variable '$strVarname' unbekannt!<br>");
      }
     }

}

$objWetter = new Wetter();
$objWetter->nTempC=25;
echo "25 °C entsprechen $objWetter->nTempF °F <br>";
$objWetter->nTempF = 90;
echo "90 °F entsprechen $objWetter->nTempC °C ";
// Ausgabe:
// 25 °C entsprechen 77 °F
// 90 °F entsprechen 32.2 °C

?>
```

Lesen Sie mehr zu den weiteren Möglichkeiten mit diesen und anderen speziellen Methoden in Kapitel 7, *Magische Funktionen*.

1.2 SimpleXML erleichtert den Umgang mit XML-Dateien

Dank SimpleXML wird der Umgang mit XML-Dateien wesentlich einfacher. Wo früher ziemliche Klimmzüge notwendig waren, nur um eine Konfigurationsdatei in XML einzulesen, reichen in PHP 5 ein paar simple Befehle:

```
<?
/*
* SimpleXML - so einfach wie sein Name
*/
  $objCD = simplexml_load_file('musik.xml');
  echo $objCD->cd[0]->titel;
?>
```

Listing 1.4 Der Name ist Programm: Die neuen SimpleXML-Funktionen vereinfachen das Arbeiten mit XML-Dateien drastisch.

Die Funktion `simplexml_load_file('musik.xml')` erzeugt aus der Datei *musik.xml* ein neues XML-Objekt und weist es `$objCD` zu.

Auf den Inhalt dieses Objekts greift die Zeile

```
echo $objCD->cd[0]->titel
```

zu. Hier ist schon die Struktur zu erkennen: Aus dem neu erzeugten XML-Objekt holt `cd[0]` den ersten Eintrag mit dem Tag `<CD>` und ermittelt genau daraus den Inhalt des Tags `<titel>`. Sprich: Die Tags werden in der Objektstruktur abgebildet und sind somit einfach zugänglich.

Mehr zu SimpleXML lesen Sie in Abschnitt 10.1, *SimpleXML*.

1.3 DOM ist jetzt besser umgesetzt

In PHP 4 war die XML-Implementierung nicht optimal. Das resultierte nicht zuletzt aus dem nicht durchgängig objektorientierten Ansatz von PHP. Mit PHP 5 ist das nun anders. Es unterstützt DOM[3], und es basiert auf der Open Source-Bibliothek libxml2. Diese gilt als stabil und zuverlässig.

```
<?php
// DOM-Beispiel
// Datei oeffnen

$objDom = new domDocument();
$objDom->load('musik.xml');

$objInterpreten = $objDom->getElementsByTagName('band');

foreach ($objInterpreten as $myInterpret)
{
  echo $myInterpret->firstChild->nodeValue."<br/>\n";
}
?>
```

Listing 1.5 Durch die neugestaltete Unterstützung des Document Object Model (DOM) hat man nun eine bewährte und standardisierte Methode zum Zugriff auf XML-Dateien.

Sie erzeugen mit `new domDocument()` ein neues DOM-Objekt. Darauf greifen Sie im weiteren Verlauf des Listings zu. Zunächst füllen Sie das Objekt mit Daten. Das erledigt der Befehl `load`, der als Parameter den zu ladenden Dateinamen mitbekommt.

3 Das Document Object Model ist eine vom World Wide Web Consortium festgelegte Programmierschnittstelle für XML- und HTML-Dokumente. Es definiert Schnittstellen und Methoden zum Erzeugen, Durchsuchen und Ändern von Inhalten.

Die Methode `getElementsByTagName` holt anhand des übergebenen Tags alle Elemente aus der XML-Datei, die das Tag `<band>` enthalten. Das Ergebnis dieser Aktion wird in `$objInterpreten` gespeichert. Hierauf greift wieder eine `foreach`-Schleife zu. Interessant wird es in der Schleife. Sie holt aus dem `<band>`-Tag den Inhalt des ersten »Kindes«. Hierbei handelt es sich um den Inhalt des Elements, den das Programm mit `nodeValue` ausgibt.

Interessant? Dann lesen Sie mehr zu DOM in Abschnitt 10.2, *Document Object Model.*

1.4 Neubeginn bei MySQL

Für Nutzer von für MySQL wurde manches anders. Einerseits darf es aus juristischen Gründen keine automatisch installierte Unterstützung mehr für diese beliebte Datenbank geben. Man kann sie allerdings recht einfach manuell nachrüsten. Andererseits wurde mit mysqli eine aufgeräumte ganz neue Funktionssammlung entwickelt. Das neue mysqli lehnt sich zwar an die bekannten Funktionen an, arbeitet aber nur mit MySQL-Versionen ab 4.1, deren neue Fähigkeiten, wie etwa das Transaktionshandling direkt unterstützt werden.

Der größte Fortschritt für OOP-Jünger ist allerdings, dass man die neue MySQL-Bibliothek auch objektorientiert verwenden kann:

```
<?php
// Beispiel mysqli
// Datenbankanfrage im OOP-Stil

// Verbindung herstellen
$objConn = new mysqli("localhost", "user", "password");
$objConn->select_db("db");
// Abfrage durchfuehren
$objResult = $objConn->query("SELECT feld1 FROM tabelle ");

// Ergebnis der Abfrage durcharbeiten
while ($row = $objResult->fetch_assoc()) {
   echo $row['feld1'] ."<br>";
}
$objConn->close();
?>
```

Wenn Sie die verbreitete Datenbank einsetzen, dann erfahren Sie in Kapitel 12, *MySQL*, mehr zu den Änderungen, die in PHP 5 auf Sie zukommen.

1.5 SQLite als Alternative

Wer mit MySQL gute Erfahrungen gemacht hat, wird sich fragen, warum mit SQLite eine neue – und bislang völlig unbekannte – Datenbank aufgenommen wurde. Doch SQLite hat es in sich: Während bei MySQL und allen anderen Datenspeichern, die man über SQL-Kommandos anspricht, ein eigener Server laufen muss, steckt diese Funktionalität im Treiber selbst. Man braucht also lediglich PHP und kann schon mit den gewohnten SQL-Befehlen Tabellen anlegen, Daten speichern und abfragen. Als Informationsspeicher legt sich SQLite eine einzige Datei an, die die gesamte Datenbasis enthält.

Bei großen Projekten, wo sehr viele Datenzugriffe gleichzeitig erfolgen, wird SQLite den etablierten Datenbanken sicher nicht das Wasser abgraben, weil ihm dann die Puste ausgeht. Aber der Funktionsumfang, den es zu bieten hat, ist erstaunlich hoch.

```
<?php
// Beispiel SQLite
// Datenbankabfrage

// Datenbankverbindung herstellen
//
$sqliteDb='/htdocs/php5/sqlite/buch.sqlite';
if (!$db = sqlite_open($sqliteDb, 0666, $err) )
   die("Fehler beim Öffnen der DB: $err");
// Abfrage starten
$query="SELECT * FROM ADRESSEN WHERE VORWAHL='089'";
$result = sqlite_query($db, $query);
// Ausgabe der Abfrageergebnisse
echo("Ausgabe der SQL-Abfrage<br><i>$query</i>:<br><br>");
echo( '<b>Name Telefon</b><br>');
while ($row = sqlite_fetch_array($result)) {
   echo( $row['name'] . '  ' );
   echo( $row['vorwahl'] . '-');
   echo( $row['telnr'] . '<br>');
}
// Verbindung schliessen
sqlite_close($db);
?>
```

Eine Einführung zu dieser interessanten Alternative finden Sie in Kapitel 13, *SQLite*.

1.6 Neue Funktionen

Nicht jede Änderung an PHP 5 bringt gleich ein neues Konzept mit sich, wie die Umstellungen bei der objektorientierten Programmierung. Es gibt auch eine ganze Reihe von nützlichen Einzelfunktionen, die das umfangreiche Arsenal von PHP weiter ausbauen.

1.6.1 Beispiel: Mail-Anhang erzeugen

So kann man mit der Funktion `convert_uuencode` binäre Daten so umwandeln, dass sie als E-Mail verschickt und von jedem üblichen Mail-Programm verarbeitet werden können. Das folgende Beispiel zeigt, wie man damit eine Datei ohne großen Aufwand versendet.

```
<?
// Beispiel - weitere neue Funktionen
// Mails mit Anhang generieren
$filename="/tmp/myfile.bin";
$handle = fopen ($filename, "r");
$contents = fread ($handle, filesize ($filename));
fclose ($handle);
$mailtext = "begin 666 myfile.bin\r\n";
$mailtext .= convert_uuencode($contents );
$mailtext .=  "end\r\n";
echo mail('sie@ihrname.de','Test UU-Anhang',$mailtext);
?>
```

1.6.2 Beispiel: Parameterliste generieren lassen

In großen Web-Anwendungen haben sich die einzelnen Skripten viel zu sagen. Die Wertübergabe von einem Programmteil zum anderen löst man meistens mit Parametern, die in der bekannten Form »`par1=x&par2=y`…« in die URL einkodiert werden. Das Zusammenbauen dieser Parameterliste kann nun PHP für Sie erledigen. Dafür sorgt die neue Funktion `http_build_query`:

```
<?
// Beispiel - neue Funktionen
// URL-Parameter aufbereiten mit http_build_query
$arrParams = array(
   'aktion' => 'kaufen',
   'kundennr'=>1234,
```

```
    'artikelnr'=>4711,
    'menge'=>3,

);
$strZielUrl = 'warenkorb.php?' . http_build_query($arrParams);
echo "<a href='$strZielUrl'>In den Warenkorb legen</a>";
?>
```

Mit dieser Funktion vermeiden Sie kryptische und ellenlange Stringkonstrukte, die man beim manuellen Zusammenbauen der URL erhält.

Weitere Neuzugänge in den bislang schon existierenden Funktionskategorien finden Sie in Kapitel 14, *Weitere neue Funktionen und Features im Überblick*.

2 OOP mit PHP 5

Die objektorientierte Programmierung ist kein Selbstzweck, sondern ein praktikabler Weg, viele Probleme beim Schreiben von Software besser zu lösen. Bei konsequenter Umsetzung spart man sich viel Mühe. Weil PHP 5 auf diesem Gebiet nun kräftig dazugelernt hat, lohnt die Beschäftigung mit diesem Thema gerade jetzt.

2.1 Der ideale Zeitpunkt zum Umstieg

Haben Sie sich schon länger Gedanken über einen Umstieg auf die objektorientierte Programmierung (OOP) gemacht? Dann probieren Sie es jetzt doch einfach.

Mit PHP können Sie Ihre bestehenden Projekte um objektorientierten Code erweitern. Planen Sie, die nächsten Features Ihrer Website in objektorientiertem PHP zu schreiben. Sie werden sehen, dass es viele Vorteile gibt. OOP bringt Sie dazu, sauberen und übersichtlichen Code zu programmieren.

Lassen Sie sich nicht von OOP verwirren; fangen Sie mit einer ganz kleinen Klasse an, und erweitern Sie diese – Sie werden sehen, dass sich die Vorteile von OOP schon nach kurzer Zeit erschließen. Und mit den neuen OOP-Fähigkeiten von PHP wird der Umstieg noch attraktiver.

Die meisten Beispiele in diesem Buch sind in der OOP-Variante von PHP geschrieben. Nur an Stellen, an denen OOP unsinnig wäre, greifen wir auf Funktionen zurück.

2.2 Grundlagen der OOP

Die objektorientierte Programmierung wird von manchen elitären Programmierern fast zu einer eigenen Kunstform erhoben. Dabei steckt dahinter im Prinzip nur eine andere Sichtweise auf Programme.[1]

Das wirklich Neue an OOP ist die Bündelung von Daten und Algorithmen zu Einheiten, den so genannten Klassen. Die Funktionen einer Klasse – zur besse-

1 Dass mit OOP kein völliges Neuland betreten wurde, sieht man zum Beispiel an den ersten Compilern für C++ – der OOP-Variante der Programmiersprache C. Diese waren lediglich als Präprozessoren ausgebildet. Das bedeutet, dass sie den objektorientierten Code in normalen C-Quellcode umgesetzt haben, der dann mit einem ganz normalen C-Compiler übersetzt wurde. Anders gesagt: Man hätte das Endergebnis auch ganz ohne OOP hinbekommen können. Nur nimmt einem die OOP-Fähigkeit einer Programmiersprache manche Arbeit ab, um sich auf das Wesentliche konzentrieren zu können.

ren Unterscheidbarkeit gegenüber der klassischen Programmierung nennt man die Funktionen dann Methoden – kümmern sich um die korrekte Behandlung der Daten, die sie in sich tragen. Als Nutzer der Klassen kann man sich also darauf konzentrieren, mit ihnen zu arbeiten.

Für die Wiederverwendbarkeit von Code ist das Konzept der Klassen ideal. Denn alles, was ihr Nutzer für den erfolgreichen Einsatz wissen muss, sind die Wirkung und das Verhalten ihrer Methoden und der nach außen hin sichtbar gemachten Variablen. Wie eine Klasse eine bestimmte Wirkung erzielt, kann demjenigen, der sie verwendet, völlig egal sein. Wichtig ist lediglich, dass man ihre »Gebrauchsanleitung« einhält, sie also mit den richtigen Daten füttert und ihre Methoden so verwendet, wie vorgesehen.

Bei richtiger Auslegung einer Klasse sind ihre Kontaktstellen zur Außenwelt nur auf das Nötigste beschränkt. Man sieht also nur das, was wichtig ist, und wird nicht durch Details verwirrt, die nur klassenintern von Interesse sind. Dazu kommt noch der Vorteil, dass sich eine Klasse fast immer leicht gegen eine neuere Version mit verbesserter Programmierung austauschen lässt – vorausgesetzt, ihr Verhalten ist gleich geblieben. Das ist so, als wenn Sie in Ihr Auto bei der nächsten Inspektion mal eben einen neuen Motor einbauen lassen könnten, der weniger verbraucht oder mehr Leistung hat.

2.3 Das neue Objektkonzept in PHP 5

Vor PHP 5 wurden Objekte, die man als Parameter in einer Funktion oder Methode eines anderen Objekts verwendete, zuerst kopiert, und dann wurde diese Kopie übergeben. Je nach Umfang des Objekts konnte dabei ein großer Zeit- und Speicherplatzbedarf entstehen. Noch entscheidender war allerdings: Man konnte in der Funktion zwar Eigenschaften des Objekts ändern, die Änderungen waren aber auf die Kopie beschränkt. Das Original-Objekt bekam davon nichts mit. Man konnte dieses Verhalten zwar durch den Einsatz des Referenz-Operators `&` ändern, aber die normale Verhaltensweise war das Kopieren von Objekten.

Die aktuelle Ausgabe von PHP verhält sich nun so wie alle ernst zu nehmenden OOP-Sprachen und arbeitet generell mit Referenzen für Objekte, überträgt also an eine Funktion nur noch ihre Adresse und nicht mehr den gesamten Inhalt.

2.4 Klonen von Objekten

Manchmal möchte man allerdings explizit eine Kopie eines Objekts anlegen, etwa um sicherzustellen, dass eine mit dem Objekt als Parameter aufgerufene Funktion die Daten wirklich unangetastet lässt.

Index