PHP
Proseminar
Web Site Building Techniques

Autoren:

Jacob, Frank
Armagan, Hamza

Inhaltsverzeichniss

1. Einsatzmöglichkeiten und Hintergründe

1.1 Was ist PHP ?

PHP ist eine Serverseitige Skriptsprache zur dynamischen Erstellung von Webseiten. Die Webseite wird sozusagen neu generiert, bevor der Benutzer die Webseite downloadet. Die Programmiersprache PHP steht für Hyptext Preprocessor und liegt als Open Source vor. Das bringt viele Vorteile mit sich. Dadurch ergibt sich die Möglichkeit das ein Entwickler die Software für kommerzielle oder nicht kommerzielle Zwecke einsetzten kann. Der Entwickler hat auch die Möglichkeit in den Quellcode reinzuschauen und eventuell den Code an eigenen Bedürfnissen anpassen.

1.2 Geschichte und Umfeld von PHP !

Die Wurzeln von PHP reichen bis ins Jahr 1985 zurück. Der damals 17 jährige Rasmus Lerdorf entwickelte in der Skriptsprache PERL eine Reihe von Funktionen zur dynamischen Erstellung von Homepages. 1994 wurde dann seine Arbeit in PHP umgetauft und die erste Version davon veröffentlicht. Im Laufe der Zeit wurde PHP ständig weiterentwickelt. Heute ist man bei PHP 5 angelangt, das z.B. einen schnelleren Sprachkern „Zend Engine 2“, try- und catch- Fehlerroutinen, neue XML – Engine und Iteratoren hat.

1.3 Was kann man mit PHP alles machen ?

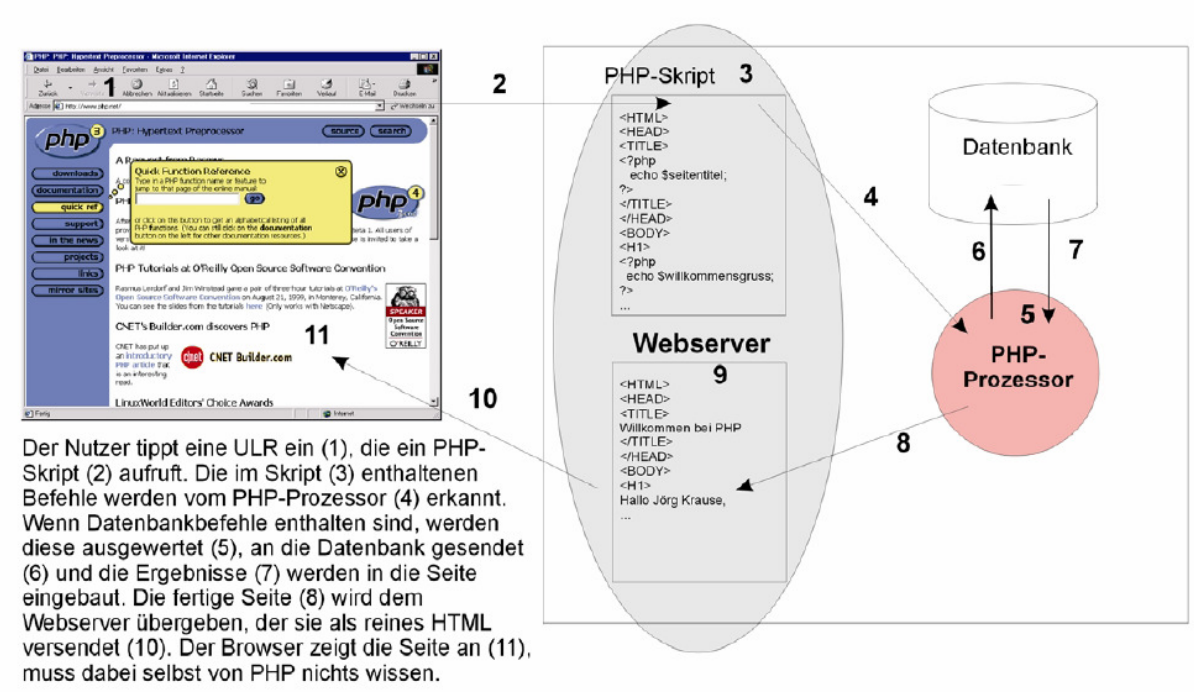
Mit der Leistungsfähigkeit von PHP hat man eine Vielfalt von Anwendungsmöglichkeiten. Dazu gehören u.a. dynamische Erstellung von Webseiten, Grafiken , Foren, Mailinglisten, Suchmaschinen, Auswertungen von HTML – Formularen, Generierung von PDF (Word, Excel, …) Dateien, Netzwerkprogrammierung (FTP, http, SMTP, POP3, ….), Datenbankzugriff usw. .

1.4 Was zeichnet PHP aus ?

Die Stabilität, gute Datenbankanbindung, Plattformunabhängigkeit, guter Support und gute Dokumentation gehören zu den meist ausprägenden Merkmalen von PHP. Nicht zu vergessen ist auch, das die Syntax von PHP ähnlich wie C / C++, Java oder JavaScript ist. Dadurch ergibt sich eine leichte Erlernbarkeit von PHP für viele Entwickler. Das wiederum erhöht die Lesbarkeit und die Wartungskosten sinken.

2. Grundlagen und Vorbereitung

2.1 Wie funktionierte PHP ?



Diese Grafik erleichtert das Verstehen von PHP. Wie schon im Diagramm beschrieben tippt der Benutzer eine URL ein. Diese muss allerdings die Eigenschaft haben, das sie mit *.php endet. Nachdem der Webserver eine Anfrage bekommt, sieht er an der Endung *.php das es sich um eine PHP Skript handelt. Demzufolge leitet der Webserver das Skript an den PHP - Prozessor weiter. Der PHP Prozessor wertet das PHP Skript aus, und führt das Programm aus, das im Skript integriert ist. Falls eine Notwendigkeit für einen Datenbankzugriff gibt, entnimmt der PHP Prozessor Daten aus der Datenbank und fügt sie in das PHP Skript ein. Das fertig generierte Skript wird wieder dem Webserver zurückgegeben, der wiederum das fertige Skript an den Browser zurückleitet.

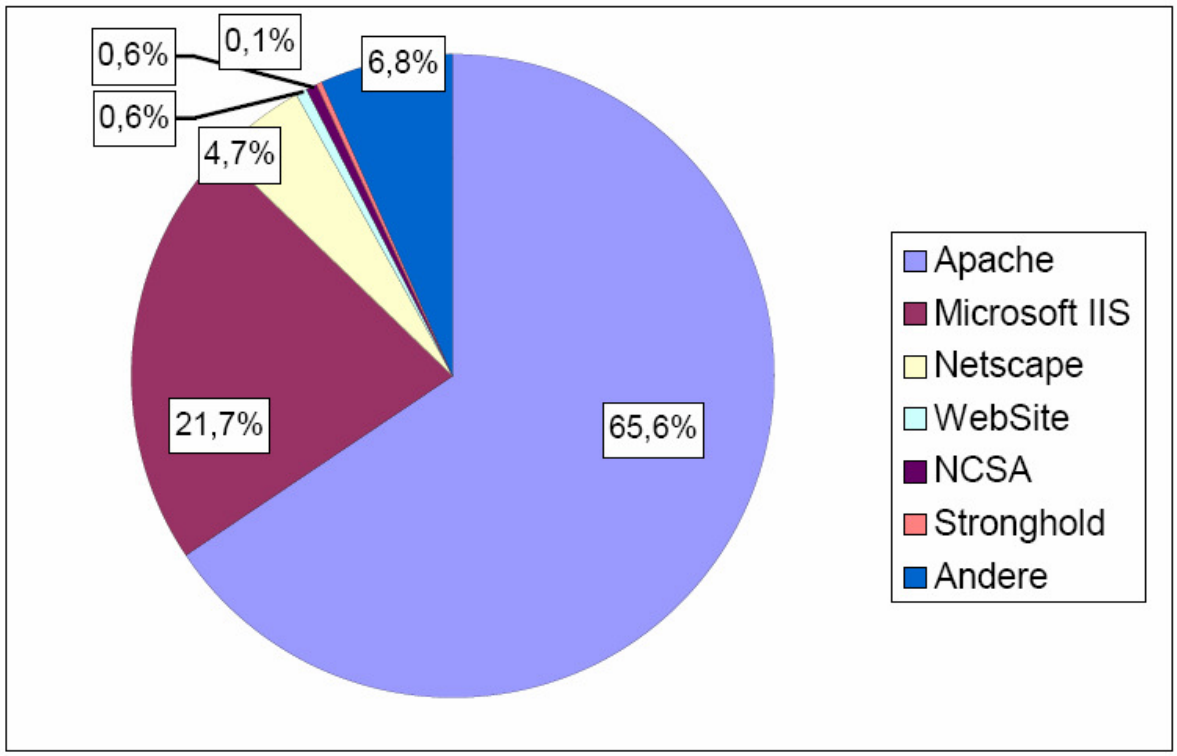
2.2 Wie installiert man PHP unter Windows 2000 / XP ?

2.2.1 Was braucht man ?

Damit man in PHP Programmieren kann braucht man PHP - Software, Texteditor, Webbrowser, Webserver und einen Datenbankserver. Dabei kann man einen beliebigen Texteditor wie Notepad, emacs, vi usw. nehmen. Für den Webbrowser hat man auch freie Wahlmöglichkeiten wie Internet Explorer, Firefox, Konqueror usw. . Als Webserver könnte zum Beispiel Apache, Microsoft IIS oder Netscape in Frage kommen. Als Datenbankserver stellen sich Hersteller wie MySql, SQLite, Oracle oder PostgrSQL in Frage.

2.2.2 Was ist ein Webserver ?

Ein Webserver leitet Skripte anhand der Dateierweiterung an die zuständigen Stellen weiter. Eine Zuständige Stelle kann zum Beispiel ein PHP Modul oder der Browser vom Client sein. Zu den meist verbreiteten Webservern gehören Apache, wie man aus dem nachfolgenden Diagramm entnehmen kann.



Quelle: E-Soft Inc. www.e-softinc.com/survey

2.2.3 Was ist ein Datenbankserver ?

Ein Datenbankserver ist ein System, dass einen konsistenten, atomaren, einfachen , dauerhaften und schnellen Zugriff auf die Daten ermöglicht. Der Datenbankserver ist sozusagen eine Schnittstelle zwischen ein PHP Modul und den Daten auf der Festplatte.

2.2.4 Was versteht man unter XAMPP ?

Die Bezeichnung XAMPP ist eine Abkürzung für eXtended Apache, MySql und PHP + Pear. Es ist sozusagen ein geschlossenes, paketorierntiertes System, indem alle Notwendige Module integriert sind. Unter dem Link http://www.apachefriends.org/en/xampp.html kann man ein Setup Programm runterladen, der Automatisch alles auf dem Rechner installiert. Wenn man PHP allerdings richtig verstehen will, ist es ratsam, alles selbst zu installieren und konfigurieren.

2.3 Was ist das File php.ini ?

Die php.ini Datei ist eine Konfigurationsdatei. Dort hat man Möglichkeiten wie den Parser zu de/aktivieren, sicherheitsrelevante Konfigurationen, Leistungsbegrenzungen, Fehlerbehandlung und Protokollierungen, Pfadeinstellungen, Dynamische Erweiterungen, Moduleinstellungen durchzunehmen.

3. PHP Einführung

3.1 Entwicklung PHP

Die Skriptsprache PHP hat folgende Entwicklung durchgemacht.

- 1985 Perl wird mit Funktionen von Rasmus Lerdorf erweitert.
- 1994 wird seine Arbeit in PHP unbenannt und die erste Version vorgestellt.
- 1995 Mitte wird PHP noch mal neu programmiert. Interpretation von HTML Formulardaten jetzt möglich. Hinzufügung von mSql und Postgres95.
- 1997 Vorstellung von PHP3. Es kommt ein echter Parser, ein Interpreter und eine grundlegende Objektorientierte Notation hinzu.
- 2000 Vorstellung von PHP4. Ein neuer schneller Sprachkern „Zend Engine“ mit vielen neuen Array Funktionen kommt hinzu.
- 2002 Ergänzung von
    - PEAR – PHP Extension and Application Repository.
        - PEAR ist ein Codearchiv für PHP – Erweiterungen und PHP – Bibliotheken.
        - Definiert Standards, um Entwickler beim Schreiben von portablem und wieder verwendbarem Code zu unterstützen.
    - PHP – GTK – Anbindung für clientseitige Programme
    - PHP - Shellskripte – das CLI (Command Line Interface) von PHP dient als Shellinterpreter

- 2004 Vorstellung von PHP5 und folgen Veränderungen wurden durchgeführt
    - Ein weiterer neuer geschriebener Sprachkern „Zend Engine2“.
    - Verbesserung der Objektorientierung
        - Als public, protected und private definierten Variablen
        - >>echte<< Instanziierung von Objekten
        - Konstruktor und Destruktor
    - Try- und catch – Fehlerroutinen
    - SQLite als eingebettete Datenbank
    - Neue XML – Engine, basierend auf libxml2, und SimpleXML als vereinfachte XML-Schnittstelle.
    - Iteratoren

3.2Vergleich mit anderen objektorientierten Programmiersprachen

Das Folgende Diagramm einen Vergleich zwischen JAVA, PHP 5 und C# angesichts der Eigenschaften in der Objektorientierung.

Feature	PHP	C#	Java
Multiple Inheritance	No	No	No
Interfaces	Yes	Yes	Yes
Operator overloading	No	Yes	No
Method overloading	No	Yes	Yes
Exception Handling	Yes	Yes	Yes
Abstract Classes	Yes	Yes	Yes
Static Class Variables	Yes	Yes	Yes
Constructors / Destructors	Yes	Yes	Yes
Polymorphism	Yes	Yes	Yes
Access Modifier (public,private)	Yes	Yes	Yes

3.3Einführung in die Programmierung

3.3.1 Kommentare

Kommentarbeispiele :

```
// ich bin ein einzeiliger Kommentar
/* und mich kann man über mehrere
   Zeilen schreiben */
```

3.3.2 Beispiel „Hallo PHP“

Einbettung in HTML (Endung .php)

```
<?                          /* Code */              ?>
<?php                       /* Bevorzugt */         ?>
<script language="php">     /* Code */      </script>
<%                          /* Code */              %>
```

Ausgaben

```
<?php  print(“Hallo PHP“);
       echo “Hallo PHP“;     ?>
```

Ein “ ; “ gehört ans Ende jeder Zeile

3.3.3 Variablen

Unterstützung von Variablen wie z.B.:

- Numerische Variablen (int, double)
- Strings
- Boolesche Variable (Inhalt true / false)

Beispiele:

```
$text   = “Das ist ein String !“;
echo $text;
echo $text,$text,$text ;
$gr     =  “groß“ ;
$kl     =  “klein“ ;
echo “Eine Limousine ist $gr . “ ;
echo “Ein Mofa ist $kl . “ ;
$i      = 10 ;
$j      = 5 ;
echo $i , “ + “ ,  $j  ,  “ = “ , $i + $j ;
```

3.3.4 Operationen

Erklärungen :

- “ + “ : Addition, $i + $j,
- “ - “ : Subtraktion, $i - $j,
- “ * “ : Multiplikation, $i * $j,
- “ / “ : Division, $i/$j,
- “% “ : Rest, $i % $j: z.B. 23 % 17 = 6 ,
- “ . “ : Verknüpfung Strings

```
$str1 = “großer“ ;   $str2 = “Kleiner“ ; echo $str1.$str2 ;
```

➔ großerKleiner

3.3.5 Post / Pre Schreibweise

Erklärungen:

$i++	erhöht $i um 1
++$i	erhöht $i ebenfalls um 1
$i--	erniedrigt $i um 1
--$i	erniedrigt $i ebenfalls um 1

3.3.6 Schleife

Beispiel while Schleife:

```
$str = " in Schleife ! <BR> " ;
$i = 0 ;

while ($i<10){
        echo $str ;
        $i++;
}
```

Beispiel for Schleife:

```
$str = " in Schleife ! <BR> " ;

for ($i=0 ; $i<10; $i++){
        echo $str ;
}
```

3.3.7 IF- Abfrage

Beispiel IF – Abfrage:

```
if($i < 0 ) {
        echo "$i ist kleiner als 0 " ;
} else {
        echo "$i ist nicht kleiner als 0";
}
```

3.3.8 Sonst

== : gleich ?
!= : ungleich ?
>= : größer oder gleich ?
<= : kleiner oder gleich ?
&& : beide Variablen true ?
|| : eine der Variable true ?

3.3.9 Funktionen

Beispiel einer Quadratsfunktion;

```
$zahl = 5 ;    $ergebnis = 0;
function quadrat($zahlIn){
        return $zahlIn * $zahlIn ;
}
$ergebnis = quadrat($zahl) ;
echo $ergebnis ;
```

4. Praxisbeispiele

4.1Grafik Rotation

```
<?php
    $Dateiname = "images/bmw.jpeg";
    $Grad = 90;

    $Grafik = imagecreatefromjpeg($Dateiname);
    $RotierteGrafik = imagerotate($Grafik, $Grad, 0);

    header('Content-type: image/jpeg');
    imagejpeg($RotierteGrafik);
?>
```

imagecreatefromjpeg($Dateiname) ➔ erzeugt ein Jpeg von $Dateinamen
imagerotate($Grafik, $Grad, 0) ➔ Rotiert das Bild $Grafik um 90 Grad.

4.2Grafik Beschriftung

```
<?php
        $Dateiname = "images/bmw.jpeg";
        $text = 'Copyright';
        $img = imageCreateTrueColor(400,200);

        $schwarz = imageColorAllocate($img,0,0,0);
        $Grafik = imagecreatefromjpeg($Dateiname);

        imageString($Grafik,5,200,10,$text,$schwarz);

        header('Content-type: image/jpeg');
        imagejpeg($Grafik);
        imageDestroy($Grafik);
?>
```

imageColorAllocate($img,0,0,0) ➔ reserviert schwarze Farbe durch Kombination 0,0,0
imagecreatefromjpeg($Dateiname) ➔ erzeugt ein Jpeg von $Dateinamen
imageString($Grafik,5,200,10,$text,$schwarz) ➔ schreibt die Variable Text auf $Grafik
imagejpeg($Grafik) ➔ Ausgabe von $Grafik

4.3Zeitmessung

```
<?php
      $start = time();
      sleep(2);
      $end = time();

      echo "Das Skript hat ".($end-$start).
      " Sekunden gebraucht um zu laden.<br>";
?>
```

$start = time() ➔ Variable $start wird mit einem Zeitwert initialisiert
sleep(2) ➔ An dieser Stelle setzt das Programm 2 Sekunden aus

4.4Speicherplatzmessung

```
<?php
      $FreierSpeicherplatz = disk_free_space("/");
      $InsgesamterSpeicherplatz = disk_total_space("/");
      $BelegtenSpeicherplatz = $InsgesamterSpeicherplatz-$FreierSpeicherplatz;

      echo"Der freie Speicherplatz beträgt: ",
         ZahlenFormatieren($FreierSpeicherplatz),
         ".<br>Der belegte Speicherplatz beträgt: ",
         ZahlenFormatieren($BelegtenSpeicherplatz),
         ".<br>Der insgesamte Speicherplatz beträgt: ",
         ZahlenFormatieren($InsgesamterSpeicherplatz),
         ".";
?>
```

disk_free_space("/") ➔ ermittelt freien Speicherplatz
disk_total_space("/") ➔ ermittelt gesamten Speicherplatz

4.5Counter

```
<?php
   if (file_exists("counter.txt")){
   $datei = fopen("counter.txt","r");
   $zaehler = fgets($datei,10);
   fclose($datei);
   }else{
   $zaehler = 0;
   }
   $zaehler++;
   $datei = fopen("counter.txt","w");
   fputs($datei,"$zaehler");
   fclose($datei);
?>
```

file_exists("counter.txt") ➔ Überprüfung ob File existiert
fopen("counter.txt","r") ➔ Öffnet das File
fclose($datei) ➔ schließt File
fgets($datei,10) ➔ liest File aus
fputs($datei,"$zaehler") ➔ schreibt in File rein

5. PHP und JavaScript

Eine Werteweitergabe von PHP zu JavaScript ist nicht direkt möglich, denn PHP liegt auf einem Server und wird auch dort interpretiert. Meist wird dabei HTML Code erzeugt und dieser wird an den Client gesendet um dann wiederum dort von dem Browser interpretiert zu werden.
JavaScript wird, wie auch der HTML Code, auf dem Client vom Browser interpretiert.
Um also einen Variablenwert von PHP an Javascript zu übermitteln, muss der herkömmliche Weg über die Http Protokolle gegangen werden, also eine typische Client-Server Anfragesituation.

PHP Variablen Inhalt an JavaScript
Um nun von PHP einen Wert zu JavaScript zu bekommen muss PHP wehrend der Interpretation auf dem Server einen Javascript Code erzeugen der den bestimmten Wert beinhaltet, z.B. eine JavaScript Zuweisung.

Beispiel 5.1: JavaScript „Hallo Welt“ Ausgabe mittels PHP

```
<?php
$content = 'Hallo Welt!';

echo '<script language="javascript" type="text/javascript">';
echo 'var jsvar = "'.$content.'!";';
echo 'alert(jsvar);';
echo "</script>";

?>
```

Erklärung:

- In diesem Beispiel soll die PHP Variable *$content* an JavaScript übergeben werden.
- Nachdem der PHP Interpreter den Code interpretiert hat bleibt folgendes übrig was dann an den Client gesendet und dort interpretiert wird:

```
<script language="javascript" type="text/javascript">
var jsvar = "Hallo Welt!";
alert(jsvar);
</script>
```

Resultat:

- Beim Client öffnet sich nun ein Fenster mit der Meldung „Hallo Welt“.

JavaScript Variablen Inhalt an PHP
Diese Richtung der Inhaltweitergabe ist etwas komplizierter, denn durch die Sicherheitskonzepte der Server-Client Kommunikation, ist der Client durch diverse Einschränkungen vor Missbrauch geschützt. So wurde unter Anderem JavaScript erheblich in seinem Handlungsspielraum eingeschränkt. Es können zum Beispiel keine Daten vom Client zum Server gelangen, ohne das der Client Benutzer den Transfer bestätigt.

Um nun einen Wer von JavaScript zu PHP zu bekommen benötigen wir ein Formular oder einen Link auf den der Benutzer klicken muss. Der Benötigte Wert wird nun entweder in der URL, oder als POST Wert an der Server übermittelt.

Beispiel 5.2: PHP „Hallo Welt“ Ausgabe mittels JavaScript

```
<?
if (isset($_GET['jsvar'])) echo $_GET['jsvar'];
?>

<script language="javascript" type="text/javascript">
var jsvar = 'Hallo Welt!';
</script>
<p>
<a href="#" onClick="window.location.href = '?jsvar='+jsvar; return
false;">Klick hier</a>
</p>
```

Erklärung:

- Das *onClick* Event des *a* Tags ruft die selbe Seite mit den nötigen GET Parametern erneut auf und übergibt so den Variablenwert.
- *isset($_GET['jsvar'])* prüft nun ob die Variable *jsvar* wirklich in der URL übergeben wurde.
- Ist dies der Fall wird mit *echo $_GET['jsvar'];* der übergebene Wert ausgegeben.

Resultat:

- Klickt der Benutzer auf den Link „Klick hier”, wird die Seite neu geladen und es ist der Schriftzug „Hallo Welt!“ zu sehen.

6. PHP und Java

Es gibt zwei Wege, um die Welten von Java und PHP zu verbinden. Einerseits besteht die Möglichkeit PHP in eine Java Servlet Umgebung zu integrieren, andererseits kann man mit der Java Extension aus PHP heraus auf Java Klassen zugreifen. Die erste der beiden genannten Möglichkeit ist wesentlich stabiler und performanter als die zweitgenannte Lösung.

Die Java Erweiterung für PHP nutzt das Object Overloading von PHP 4 um auf Java Klassen zuzugreifen. Die Java Virtual Machine (JVM) wird hierbei über das Java Native Interface (JNI) erzeugt. Eine detaillierte Installationsbeschreibung kann der Datei „`php4/ext/java/README`“ entnommen werden.

Beispiel 6.1: Zugriff auf Java Klassen aus PHP heraus

```
<?php

// In PHP Instanz der Java Klasse java.lang.System erzeugen
$system = new Java('java.lang.System');

// Zugriff auf die Objekteigenschaften
print 'Java version='.$system->getProperty('java.version').' <br>';
print 'Java vendor=' .$system->getProperty('java.vendor').'  <br>';
print 'OS='.$system->getProperty('os.name').' '.
           $system->getProperty('os.version').' on '.
           $system->getProperty('os.arch').' <br>';

// java.util.Date Beispiel
$formatter = new Java('java.text.SimpleDateFormat',
                      "EEEE, MMMM dd, yyyy 'at' h:mm:ss a zzzz");

print $formatter->format(new Java('java.util.Date'));

?>
```

Beispiel 6.2: Zugriff auf Java AWT aus PHP heraus

```
<?php

// Dieses Beispiel kann nur als CGI ausgeführt werden.

$frame  = new Java('java.awt.Frame', 'PHP');
$button = new Java('java.awt.Button', 'Hallo Welt!');

$frame->add('North', $button);
$frame->validate();
$frame->pack();
$frame->visible = True;

$thread = new Java('java.lang.Thread');
$thread->sleep(10000);

$frame->dispose();
?>
```

Erklärung:

- *new Java()* erzeugt eine Instanz der angegebenen Java Klasse, falls ein geeigneter Konstruktur gefunden wird. Für den Zugriff auf Klassen mit statischen Methoden benötigt man keinen Parameter.
- Beim Zugriff auf Mitglieder (Members) einer Instanz wird zunächst nach Bean Eigenschaften gesucht, danach erst nach öffentlichen (public) Eigenschaften. Mit anderen Worten: Es wird zuerst versucht werden, zum Beispiel *print $date.time* als *$date.getTime()* zu interpretieren, danach erst wird *$date.time* versucht werden.
- Der Zugriff auf statische und Instanzeigenschaften erfolgt mit der selben Syntax.
- Das Auslösen einer Java Exception resultiert in einer PHP Warnung und einem NULL Ergebnis.
- Traditionell können Arrays und Hashes in PHP weitestgehend als gleichwertig angesehen werden. Beim Austausch dieser Datentypen zwischen PHP und Java ist zu beachten, dass Hashes in PHP nur mit Integer- und Stringwerten indiziert werden können und Arrays von primitiven Datentypen in Java nicht dünn besetzt sein dürfen. Ferner ist zu beachten, dass diese Datenstrukturen by-Value übergeben werden, was ungünstig in Bezug auf Speicher- und Zeitbedarf sein kann.

Das PHP 4 Servlet SAPI Modul baut auf dem von der Java Extension zur Verfügung gestellten Mechanismus auf, allerdings wird der PHP Prozessor hierbei von einer Servlet Engine, wie zum Beispiel Apache Jakarta / Tomcat, ausgeführt. Dies führt zu einer wesentlich höheren Stabilität und besserer Performance als der umgekehrte Weg, wie in die Java Extension an sich anbietet. Dies kommt daher, dass die Servlet Engine sich um das Pooling und die Wiederverwendung von Java Virtual Machines (JVMs) kümmert. Eine detaillierte Anleitung zur Integration von PHP in eine Servlet Engine entnehmen Sie bitte der Datei `php4/sapi/README`. Bemerkungen:

- Das Servlet SAPI Modul sollte zwar auf jeder Java Servlet Engine funktionieren, wurde aber bislang nur in Verbindung mit Apache Jakarta/Tomcat getestet. Meldungen von Problem - aber auch von Erfolgserlebnissen - bei der Verendung mit anderen Servlet Engines werden dankbar aufgenommen.
- PHP hat die Angewohnheit das aktuelle Arbeitsverzeichnis (CWD) zu ändern. sapi/servlet wird versuchen, den alten Zustand wieder herzustellen. Während PHP läuft kann es sein, dass die Servlet Engine kann Klassen aus dem CLASSPATH finden kann, die über relative Pfade angesprochen werden.

7. Objektorientierte Programmierung

In PHP können eigene Klassen erstellt werden. Dabei muss aber die PHP Version beachtet werden, da die Syntax in den unterschiedlichen Versionen stark variiert. Grundsätzlich gilt aber das Gesetz der Abwertskompatibilität, weshalb es immer zu empfehlen ist für die neueren PHP Versionen zu programmieren.

Grundsätzlicher Aufbau einer PHP Klasse

```
<?php
class [Klassenname] extends [Name der Vaterklasse]
{

      [Eigenschaften]

      [Konstruktor und Destruktor]

      [Methoden]

};
?>
```

Erklärung:

- **[Klassenname]** muss ein Text sein der nicht mit einer Ziffer beginnt und der keine Sonderzeichen wie z.B. Leerzeichen oder "!" enthält.
- **[Name der Vaterklasse]** beschreibt die Klasse deren Inhalt an diese (Sohn)Klasse vererbt werden soll.
- **[Eigenschaften]** sind die Variablen die für dieses Objekt gelten sollen.
  Diese sind in PHP <= 4 alle von Außerhalb zugänglich (public) und werden mit einem vorangestelltem *var* deklariert (z.B. *var $name;*).
- **[Konstruktor und Destruktor]** sind die zwei Sondermethoden die an Events geknüpft sind und beim Anlegen, bzw. Löschen des Objekts durchlaufen werden.
  In PHP <=4 trägt der Konstruktor immer den Namen der Klasse und der Destruktor trägt den Namen der Klasse plus einer vorgehängten Tilde („~"), also ~[Klassenname].
  In PHP >= 5 heißt der Konstruktor immer *__construct* und der Destructor *__destruct*.
- **[Methoden]** sind die Funktionen des Objektes.
  Auch Diese sind in PHP <= 4 alle ausnahmslos von Außerhalb des Objektes zugänglich (public).

Beispiel 7.1: „Hallo Welt“ Ausgabe mittels einer PHP <= 4 Klasse

```
<?php

class Klasse

{

	# Eigenschaften



	var $eigenschaft1;

	var $eigenschaft2;

	# Konstruktor und eigener "Destruktor"

	function Klasse()

	{

		$this->eigenschaft1 = "";

		$this->eigenschaft2 = "";

	}

	function Destroy()

	{

		echo $this->eigenschaft1.$this->eigenschaft2;

		unset($this);

	}

	# Methoden

	function Methode1 ($value)

	{

		$this->eigenschaft1 = $value;

	}

	function Methode2 ($value)

	{

		$this->eigenschaft2 = $value;
```

```
      }

};

$object = new Klasse();

$object->eigenschaft1 = 'Hallo';

$object->eigenschaft2 = ' Welt!';

$object->Methode1('Hallo');

$object->Methode2(' Welt!');

$object->Destroy();

?>
```

Erklärung:

- Durch die Zeile *$object = new Klasse();* wird das Objekt aus der Klasse *Klasse* erzeugt.
- *$object->eigenschaft1* und *$object->eigenschaft1* sind die öffentlichen Eigenschaften des Objektes, werden im Konstruktor *Klasse* initialisiert und bekommen in unserem Beispiel die Texte „Hallo“ und „ Welt!“ zugewiesen. Es gibt keine Möglichkeit diese Eigenschaften vor öffentlichem Zugriff zu schützen.
- *$object->Methode1* und *$object->Methode2* sind die öffentlichen Methoden des Objekts. *$object->Methode1* weist der ersten Eigenschaft einen Wert zu und *$object->Methode2* weist der zweiten Eigenschaft einen Wert zu.
- *$object->Destroy* ist ebenfalls eine öffentlich zugängliche Methode die zum Einen die Inhalte der beiden Eigenschaften ausgeben soll und zum Anderen das Objekt aus dem Speicher löscht.

Resultat:

- Im Browser erscheint der Text „Hallo Welt!“.

Beispiel 7.2: Bessere „Hallo Welt“ Ausgabe mittels einer PHP >= 5 Klasse

```
<?php

class Klasse

{

      # Eigenschaften

      private $eigenschaft1 = "";

      private $eigenschaft2 = "";
```

```
      # Konstruktor und Destruktor
      public function __construct()
      {
      }
      public function __destruct()
      {
            echo $this->eigenschaft1.$this->eigenschaft2;
      }
      # Methoden
      public function Methode1 ($value)
      {
            $this->eigenschaft1 = $value;
      }
      public function Methode2 ($value)
      {
            $this->eigenschaft2 = $value;
      }
};
$object = new Klasse();
# $object->eigenschaft1 = 'Hallo';  // Nicht möglich, weil private
# $object->eigenschaft2 = ' Welt!'; // Nicht möglich, weil private
$object->Methode1('Hallo');
$object->Methode2(' Welt!');
?>
```

Erklärung:

- Durch die Zeile *$object = new Klasse();* wird das Objekt aus der Klasse *Klasse* erzeugt.

- *$object->eigenschaft1* und *$object->eigenschaft1* sind private Eigenschaften des Objektes und werden direkt (ohne Konstruktor) in der Klasse initialisiert. Auf diese Eigenschaften kann von außerhalb des Objektes **nicht** zugegriffen werden!
- *$object->Methode1* und *$object->Methode2* sind die öffentlichen Methoden des Objekts. *$object->Methode1* weist der ersten Eigenschaft einen Wert zu und *$object->Methode2* weist der zweiten Eigenschaft einen Wert zu.
- Ein expliziter Aufruf des Destructors ist in PHP >= 5 nicht notwendig, da dieser bei Beendigung des Skripts automatisch aufgerufen wird (Garbage Collector).

Resultat:

- Im Browser erscheint der Text „Hallo Welt!“.

Übersicht über die Erneuerungen in der Objektorientierung ab PHP 5

- Zugriffsbeschränkungen (*public, private, protected*) von Eigenschaften und Methoden sind jetzt möglich.
- Neu sind auch Destruktoren (*__destruct*) die beim vernichten des Objektes aufgerufen werden.
- Es gibt jetzt zwei spezielle eventgesteuerte Methoden *__get* und *__set*, die bei einer Zuweisung oder beim auslesen einer Mitgliedsvariable aufgerufen werden.
- Für *__get* und *__set* kann auch der universelle Methodenaufruf *__call* verwendet werden.
- Interfaces (Schnittstellen) zum verfeinerten Umgang mit Bibliotheken.
- Abstrakte Klassen und Mitglieder.
- Finale Klassen die nicht weiter vererbt werden können.
- Statische Mitglieder die nur einmal pro Objekttyp erzeugt werden und dann für alle Objekte gelten.

TIPP: Weitere Beispiele und fertige Klassen gibt es bei www.phpclasses.org.

8. MySQL und PHP

MySQL ist eine Open Source Datenbanken und wird in allen PHP Versionen <= 4 als Standard Datenbank verwendet, d.h. nach der Installation von PHP <= 4 war mittels der PHP MySQL-Befehle ein Zugriff direkt möglich. Das hat sich in den Version >= 5 geändert. Hier müssen die MySQL-Funktionalitäten erst durch Modifikation der *php.ini* aktiviert werden.

Beispiel 8.1: Einfacher Datenbankzugriff

```
<?php

# Verbindung zur Datenbank herstellen

$db = mysql_connect("localhost", "tester", "test") or die("Es konnte keine
Verbindung zur Datenbank hergestellt werden!");

mysql_select_db("proseminar", $db);

# Abfrage starten

$sql = "SELECT * FROM user ORDER BY importance";

$result = mysql_query ($sql, $db) or die("Die Abfrage $sql konnte nicht
durchgeführt werden!");

# Den ersten Datensatz als Objekt holen

$user = mysql_fetch_object($result);

# Speicher freigeben

mysql_free_result($result);

# Ergebnis ausgeben

echo 'Der wichtigste Benutzer ist '.$user->forename.' '.$user->name.'.';

# Datenbankverbindung schließen

mysql_close($db);

?>
```

Erklärung:

- Mit *$db = mysql_connect(…)* wird eine Verbindung zur MySQL Datenbank hergestellt und in *$db* gespeichert. „localhost“ ist hier der Datenbankserver (in diesem Falle der Localhost). „tester“ und „test“ sind der User und das Passwort für die Datenbank.

- *mysql_query(…)* führt eine Abfrage durch und liefert einen *result* der die Ergebnismenge beinhaltet.
- *mysql_fetch_object(…)* liefert aus dem *result* den gewünschten Datensatz als Objekt. Die Feldnamen sind bei diesem Objekt die Eigenschaften über die man auf die Inhalte zugreifen kann.
- *mysql_free_result(…)* gibt den belegten Speicher für die Ergebnismenge frei, so das er wieder anderweitig verwendet werden kann.
- *mysql_close(…)* trennt die Datenbankverbindung und löscht alle Verbindungsdaten aus dem Speicher.

Resultat:

- Es wird der Benutzer ausgegeben der die höchste Wichtigkeit (importance) hat.

Weitere Möglichkeiten des Datenbankzugriffs

Eine weitere Möglichkeit eines Datenbankzugriffs bietet ein ODBC (Open Database Connectivity) Treiber. Dieser muss zunächst auf dem PHP Server installiert werden. Einen ODBC Treiber für die MySQL Datenbank kann man auf der Website „dev.mysql.com“ herunterladen. Ist der Treiber fertig eingerichtet, kann mittels der PHP ODBC Befehle über ODBC auf die MySQL Datenbank zugegriffen werden.

Beispiel 8.3: Einfacher Datenbankzugriff über ODBC

```
<?php

# Verbindung zur Datenbank herstellen
$db = odbc_connect('testdb', 'tester', 'test') or die("Es konnte keine
ODBC Verbindung zur Datenbank hergestellt werden!");

# Abfrage starten
$sql = "SELECT * FROM user ORDER BY importance";
$result = odbc_exec($db, $sql) or die("Die Abfrage $sql konnte nicht
durchgeführt werden!");

# Den ersten Datensatz als Objekt holen
$user = odbc_fetch_object($result, 0);

# Speicher freigeben
odbc_free_result($result);

# Ergebnis ausgeben
echo 'Der wichtigste Benutzer ist '.$user->forename.' '.$user->name.'.';

# Datenbankverbindung trennen
odbc_close($db);

?>
```

Erklärung:

- Mit *$db = odbc_connect(…)* wird eine Verbindung zur Datenbank über ODBC hergestellt und in *$db* gespeichert. „testdb“ ist hier die DSN, welche z.B. in Windows

Betriebssystemen in der Windows-Systemsteuerung vergeben werden kann. „tester“ und „test“ sind der User und das Passwort für die Datenbank.
- *odbc_exec(…)* führt eine Abfrage durch und liefert einen *result* der die Ergebnismenge beinhaltet.
- *odbc_fetch_object(…)* liefert aus dem *result* den gewünschten Datensatz als Objekt. Die Feldnamen sind bei diesem Objekt die Eigenschaften über die man auf die Inhalte zugreifen kann.
- *odbc_free_result(…)* gibt den belegten Speicher für die Ergebnismenge wieder frei so das er wieder anderweitig verwendet werden kann.
- *odbc_close(…)* trennt die Datenbankverbindung und löscht alle Verbindungsdaten aus dem Speicher.

Resultat:

- Es wird der Benutzer ausgegeben der die höchste Wichtigkeit (importance) hat.

9. Beispiel eine PHP Datenbankanwendung

Abschließend wird im folgenden Beispiel eine Anwendung vorgestellt, die einen typische Einsatz von PHP zum dynamisieren einer Website zeigt.
Hierzu wird eine Personenverwaltung erstellt die im Kern aus HTML Code besteht, die Datenbankzugriffe und die dynamischen Elemente wurden mit PHP realisiert. Auch Javascript wurde verwendet. Jedoch haben die verwendeten Javascript Codes keine Auswirkung auf die Funktionalität der Personenverwaltung, sondern dienen lediglich der Steigerung der Benutzerfreundlichkeit.

Setup (personen_setup.xpl.php): Die verwendete Setup-Datei, die in alle Seiten der Anwendung eingebunden ist.

```
<?php

$db = mysql_connect("localhost", "user", "passwort") or die("Es konnte
keine Verbindung zur Datenbank hergestellt werden!");

mysql_select_db("proseminar", $db);

?>
```

Erklärung:

- Die Setup-Datei dient in erster Linie der Zentralisierung häufig eingesetzter Initialisierungen, Zugriffe oder Funktionalitäten. Sie wird in jede .php Datei der Anwendung eingebunden.
- In diesem Falle wird nur der Zugriff auf die Datenbank hergestellt. Eine weiterer sinnvoller Einsatz dieser Setup-Datei wäre z.B. die Session-Initialisierung oder die Definition globaler Konstanten die häufig verwendet werden.

Übersicht (personen.xpl.php): Übersicht aller Datensätze mit Datensatzoperationen

```
<? require('personen_setup.xpl.php'); ?>

<?

// Hilfsfunktion

function ne($text, $empty_value){ if (empty($text)) return $empty_value;
else return $text; };



// Jobs abarbeiten

if (strtolower($_GET['job']) == 'delete')

{

      if ($_GET['id'] > 0)

      {

            $sql = "DELETE FROM personen WHERE id = ".$_GET['id'];

            $result = mysql_query ($sql, $db) or die("Die Löschabfrage
$sql konnte nicht durchgeführt werden!");

      };

};



// Daten holen

$data = array();

$sql = "SELECT * FROM personen ORDER BY vorname";

$result = mysql_query ($sql, $db) or die("Die Abfrage $sql konnte nicht
durchgeführt werden!");

while ($dataset = mysql_fetch_array($result, MYSQL_ASSOC))

{

      $data[] = $dataset;

};
```

```
// Speicher freigeben

mysql_free_result($result);



// Datenbankverbindung schließen

mysql_close($db);

?>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />

<title>Personenverwaltung</title>

</head>



<body>

<h1>Personenverwaltung</h1>

<table width="100%" border="1" cellpadding="4" cellspacing="0"
bordercolor="#CCCCCC">

  <tr>

    <td width="80" height="30" align="center" bgcolor="#CCCCCC"><a
href="personen_eingabe.xpl.php?job=new">[Neu]</a></td>

    <th align="left" bgcolor="#CCCCCC">Vorame</th>

    <th align="left" bgcolor="#CCCCCC">Nachname</th>

    <th width="100" align="left" bgcolor="#CCCCCC">Anrede</th>

    <th width="100" align="left" bgcolor="#CCCCCC">Titel</th>

    <th width="120" align="left" bgcolor="#CCCCCC">Email</th>

    <th width="120" align="left" bgcolor="#CCCCCC">Telefon</th>

  </tr>

<?

for ($i=0; $i < count($data); $i++)
```

```
{

?>

  <tr onMouseOver="this.style.background='#FFFF00';"
onMouseOut="this.style.background='#FFFFFF';">

    <td align="center">

            <a href="personen_eingabe.xpl.php?job=edit&id=<? echo
$data[$i]['id']; ?>">[Bearbeiten]</a>

            <br>

            <a href="personen.xpl.php?job=delete&id=<? echo
$data[$i]['id']; ?>" onClick="return confirm('Diese Person wirklich
löschen?');">[L&ouml;schen]</a>

      </td>

    <td><? echo ne($data[$i]['vorname'], ' '); ?></td>

    <td><? echo ne($data[$i]['nachname'], ' '); ?></td>

    <td><? echo ne($data[$i]['anrede'], ' '); ?></td>

    <td><? echo ne($data[$i]['titel'], ' '); ?></td>

    <td><? echo ne($data[$i]['email'], ' '); ?></td>

    <td><? echo ne($data[$i]['telefon'], ' '); ?></td>

  </tr>

<?

};

?>

</table>

</body>

</html>
```

Erklärung:

- Die Datei „personen_setup.xpl.php“ wird eingebunden. Kann diese aus irgend einem Grund nicht eingebunden werden bricht der PHP Interpreter mit einer Fehlermeldung ab.
- Es wird eine Tabelle dargestellt mit allen bislang eingetragenen Personen, sortiert nach dem Vornamen.
- Durch einen Klick auf **[Neu]** wird die Seite zur Neuanlage eines Datensatzes geöffnet.

- Durch einen Klick auf **[Bearbeiten]** in der jeweiligen Datenzeile wird die Seite zur Bearbeitung eines Datensatzes geöffnet.
- Durch einen Klick auf **[Löschen]** in der jeweiligen Datenzeile wird zunächst gefragt ob der Datensatz wirklich gelöscht werden soll. Nach der Bestätigung durch einen Klick auf **[OK]** wird dann der Datensatz gelöscht. Diese Bestätigungsabfrage wurde mit Javascript realisiert und dient der Vermeidung versehendlicher Löschungen.
- Durch bewegen des Mauszeigers über eine Datenzeile, wird diese mit Javascript farblich hervorgehoben um dem Benutzer eine bessere Übersicht zu verschaffen.

Eingabe (personen_eingabe.xpl.php): Bearbeiten und Neuanlage eines Datensatzes

```
<?php

<? require('personen_setup.xpl.php'); ?>

<?

$javascript_code = "";



// Jobs abarbeiten

if (strtolower($_GET['job']) == 'edit')

{

      $page_title = 'Datensatz bearbeiten';

      if ($_GET['id'] <= 0) die('Es wurde keine Datensatz-ID übergeben!');

      if ($_POST['execute'])

      {

            // SQL zusammenbauen

            foreach($_POST['data'] as $name=>$value) $fields[] = "$name =
'$value'";

            $fields = implode(',', $fields);

            $sql = "UPDATE personen SET $fields WHERE id = ".$_GET['id'];

            $result = mysql_query ($sql, $db) or die("Die Abfrage $sql
konnte nicht durchgeführt werden!");

            $javascript_code = '<script language="javascript"
type="text/javascript">alert(\'Der Datensatz wurde erfolgreich
bearbeitet\');</script>';

      }
```

```
	else

	{

		// Zu bearbeitenden Datensatz holen

		$data = array();

		$sql = "SELECT * FROM personen WHERE id = ".$_GET['id'];

		$result = mysql_query ($sql, $db) or die("Die Abfrage $sql
konnte nicht durchgeführt werden!");

		while ($dataset = mysql_fetch_array($result, MYSQL_ASSOC))

		{

			$_POST['data'] = $dataset;

		};

	}

}

else if (strtolower($_GET['job']) == 'new')

{

	$page_title = 'Datensatz anlegen';



	if ($_POST['execute'])

	{

		// SQL zusammenbauen

		$fieldnames = implode(',', array_keys($_POST['data']));

		$fieldvalues = '"'.implode('","',
array_values($_POST['data'])).'"';

		$sql = "INSERT INTO personen ($fieldnames) VALUES
($fieldvalues)";

		$result = mysql_query ($sql, $db) or die("Die Abfrage $sql
konnte nicht durchgeführt werden!");

		$javascript_code = '<script language="javascript"
type="text/javascript">alert(\'Der Datensatz wurde erfolgreich
angelegt\');</script>';

		$_POST['data'] = array();
```

```
      };

}

else

{

      die('Es wurde keine Datenoperation ausgewählt!');

};

// Speicher freigeben

mysql_free_result($result);

// Datenbankverbindung schließen

mysql_close($db);

?>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />

<title>Personenverwaltung - <? echo $page_title; ?></title>

</head>

<body>

<h1>Personenverwaltung - <? echo $page_title; ?></h1>

<form name="form1" method="post" action="">

  <table width="500" border="1" cellpadding="4" cellspacing="0"
bordercolor="#CCCCCC">

    <tr>

      <td bgcolor="#CCCCCC">Anrede / Titel </td>

      <td>

            <select name="data[anrede]">

          <option value="Herr" <? if ($_POST['data']['anrede']=='Herr')
echo 'selected'; ?>>Herr</option>

          <option value="Frau" <? if ($_POST['data']['anrede']=='Frau')
echo 'selected'; ?>>Frau</option>
```

```
        </select>

            <select name="data[titel]">

          <option value="" <? if ($_POST['data']['titel']=='') echo
'selected'; ?>>Ohne Titel</option>

          <option value="Dr." <? if ($_POST['data']['titel']=='Dr.') echo
'selected'; ?>>Dr.</option>

          <option value="Prof." <? if ($_POST['data']['titel']=='Prof.')
echo 'selected'; ?>>Prof.</option>

        </select>

</td>

    </tr>

    <tr>

      <td width="100" bgcolor="#CCCCCC">Vorname</td>

      <td><input name="data[vorname]" type="text" value="<? echo
$_POST['data']['vorname']; ?>" size="30" maxlength="150"></td>

    </tr>

    <tr>

      <td bgcolor="#CCCCCC">Nachname</td>

      <td><input name="data[nachname]" type="text" value="<? echo
$_POST['data']['nachname']; ?>" size="30" maxlength="150"></td>

    </tr>

    <tr>

      <td bgcolor="#CCCCCC">Email</td>

      <td><input name="data[email]" type="text" value="<? echo
$_POST['data']['email']; ?>" size="30" maxlength="150"></td>

    </tr>

    <tr>

      <td bgcolor="#CCCCCC">Telefon</td>

      <td><input name="data[telefon]" type="text" value="<? echo
$_POST['data']['telefon']; ?>" size="30" maxlength="150"></td>

    </tr>
```

```
    <tr>

      <td bgcolor="#CCCCCC">Fax</td>

      <td><input name="data[fax]" type="text" value="<? echo
$_POST['data']['fax']; ?>" size="30" maxlength="150"></td>

    </tr>

    <tr>

      <td bgcolor="#CCCCCC">Adresse</td>

      <td><textarea name="data[adresse]" cols="50" rows="4"><? echo
$_POST['data']['adresse']; ?></textarea></td>

    </tr>

    <tr>

      <td bgcolor="#CCCCCC">Bemerkungen</td>

      <td><textarea name="data[bemerkungen]" cols="50" rows="4"><? echo
$_POST['data']['bemerkungen']; ?></textarea></td>

    </tr>

    <tr>

      <td bgcolor="#CCCCCC"> </td>

      <td><input name="save" type="submit" id="save" value="Speichern">

      <input name="execute" type="hidden" id="execute" value="1">

      <input type="reset" name="Submit" value="Zur&uuml;cksetzen"></td>

    </tr>

  </table>

</form>

<form name="form2" method="post" action="personen.xpl.php">

  <input name="back" type="submit" id="back" value="Zur&uuml;ck zur
&Uuml;bersicht">

</form>

</body>

</html>

<? echo $javascript_code; ?>
```

```
?>
```

Erklärung:

- Auch hier wird wieder die Datei „personen_setup.xpl.php“ eingebunden. Kann diese aus irgend einem Grund nicht eingebunden werden bricht der PHP Interpreter mit einer Fehlermeldung ab.
- Diese Seite kann mit 2 Parametern gestartet werden. Wir kein Parameter übergeben wird der Seitenaufbau mit der Meldung „Es wurde keine Datenoperation ausgewählt!“ abgebrochen. Der erste Parameter *$job* legt die Datensatzoperation fest. Die möglichen Werte sind hier „edit“ für Datensatzbearbeitung und „new“ für eine Neuanlage. Der zweite Parameter *$id* entspricht der ID des Datensatzes in der Datenbank und wird nur bei einer Datensatzbearbeitung benötigt. Mögliche Werte sind alle ganze Zahlen größer 0.
- In die Eingabefelder können die Personendaten eingetragen werden. Anführungszeichen werden nicht unterstützt und führen zu einem unbehandelten Fehler. Die Vermeidung dieses Fehlers wurde in diesem Zusammenhang nicht als Teil der Aufgabenstellung betrachtet und auf kosten der Lesbarkeit des Codes weggelassen.
- Durch einen Klick auf **[Speichern]** oder durch drücken der Taste **[Enter]** werden die eingetragenen Daten in der Datenbank gespeichert. Bei der Neuanlage eines Datensatzes wird das Formular nach dem speichern geleert und es kann ein weiterer Datensatz eingetragen werden. Das erfolgreiche Speichern wird mittels Javascript in einer Popup-Box bestätigt.
- Durch einen Klick auf **[Zurücksetzen]** werden die Eingabefelder auf den Wert nach dem letzten Speichern zurückgesetzt.

Durch einen Klick auf **[Zurück zur Übersicht]** gelangt man wieder zur Übersichtsseite und alle nicht gespeicherten Eingaben gehen verloren.

10.Quellen

Php4 - dynamische Webauftritte Professionell realisieren
Markt + Technik
Egon Schmidt und Christian Cartus
ISBN 3-8272-5877-4

PHP 4 - Grundlagen und Profiwissen
Hanser
Jörg Krause
3. überarbeitete Auflage
ISBN 3-446-22234-0

PHP 5 - Grundlagen und Profiwissen
Hanser
Jörg Krause
2. Auflage
ISBN 3-446-40334-5

PHP 5 - Das Update
Hanser
Jörg Krause
ISBN 3-446-22949-3

Einsteigen und durchstarten mit PHP 5
dpunk.verlag
Hakan Kücükyilmaz - Thomas M. Haas – Alexander Merz
ISBN 3-89864-236-4
---
www.php.net
---
www.phpclasses.org
