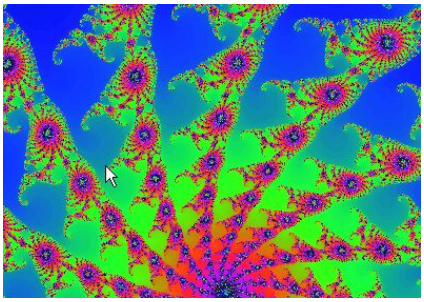
Datenbanken

10. Web-Anwendungen mit PHP und MySQL



Fachhochschule Darmstadt
Fachbereich Informatik

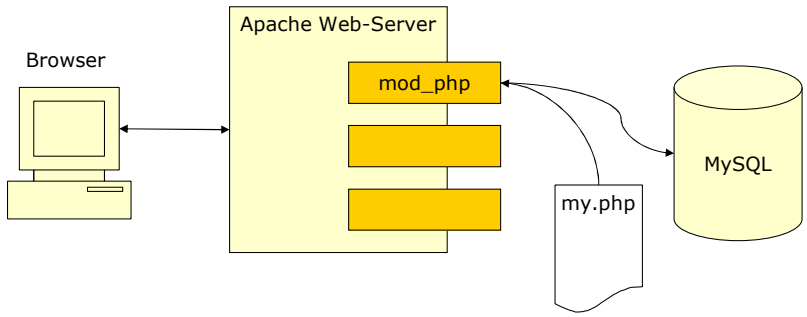
Dr. Peter Nevermann

Rückblick

- **Funktionale Abhängigkeit (X→Y)**
  - Eigenschaft des Schemas, nicht einer Relation
  - Ist eine Integritätsregel (Vorgabe aus der Miniwelt)
- **Armstrong's Inferenzregeln**
  - Vollständiges Regelsystem zur Ableitung von Abhängigkeiten
- **Normalformen (Codd)** **[Maß für die Güte eines Schemas]**
  - 1NF: Alle Attribute sind einfach (d.h. weder gegliedert noch multipel)
  - 2NF: 1NF und keine partiellen Abhängigkeiten von Schlüsseln
  - 3NF: 2NF und keine transitive Abhängigkeiten von Schlüsseln
  - BCNF (Boyce-Codd): Alle nicht-trivialen Abhängigkeiten gehen von Schlüsseln aus
- **Normalisieren = Zerlegen (R → R1 und R2)**
  - Verlustfrei ($R = R_1 \bowtie R_2$)
  - Redundanzreduzierend
  - Abhängigkeitserhaltend

XAMPP von ApacheFriends

- **Version 1.4.13 mit**
  - Apache 2.0.53
  - MySQL 4.1.11
  - PHP 5.0.4
  - ... und mehr (MiniPerl, OpenSSL, phpMyAdmin, ...)



Mit XAMPP von ApacheFriends (http://www.apachefriends.de) ist ein Paket gegeben, das MySQL und den Apache Web-Server mit PHP-Modul beinhaltet. Aufgrund der guten Unterstuetzung fuer MySQL in PHP ist diese Kombination weitverbreitet im Einsatz fuer die Erstellung datenbankbasierter Web-Anwendungen. Für die Installation lädt man im einfachsten Fall die ZIP Distribution von der PHP Homepage (z.B. xampp-win32-1.4.13.zip oder xampplite-win32-1.4.13.exe) herunter und entpackt den Inhalt direkt unter C:\, so daß ein Verzeichnis *c:\xampp* entsteht. Wählt man ein anderes Installationsverzeichnis (z.B. *d:\tools\xampp*), so muß man einmalig die Batch-Datei *setup_xampp.bat* ausfuehren, um die Umgebung korrekt zu setzen.

PHP ist eine *server-seitige Skript-Sprache* zur Erzeugung von *dynamischen* Web-Inhalten. PHP ist ein Open-Source Produkt und wurde 1994 von Rasmus Lerdorf erfunden. Die Abkürzung stand ursprünglich für "Personal Homepage Tool". Inzwischen hat PHP eine enorme Verbreitung ereicht und hat ab Version 3 (1998) längst Einzug im professionelle Bereiche gehalten. Die Abkürzung steht mittlerweile für "PHP Hypertext Processor". Die derzeit aktuelle Version ist 4, PHP 5 ist im kommen, welche u.a. objektorientierte Programmierung mit sich bringt. PHP's Homepage ist http://www.php.net.

In diesem Vorlesungs-Skript befassen wir uns hauptsächlich mit PHP 4.

Skriptsprachen im Web-Umfeld dienen generell der Erzeugung von *dynamischen* Web-Inhalten. In der Regel wird der Skript-Code in HTML-Seiten eingebunden: dazu dienen bestimmte Markierungen wie z.B. <% .... %>, <? ... ?> oder <script language ="xyz"> ... </script> die anzeigen, wo Skript-Code startet bzw. endet. Dabei wird durch die Ausführung des Skript-Codes der dynamische Anteil der HTML-Seite erzeugt (oder generiert), z.B. der Inhalt einer Tabelle der sich aus Datensätzen aus einer Datenbank ergibt.

Bei Web-Skripten unterscheidet man zwischen client- und server-seitigen Skripten. Client-seitige Skripte (Sprachen: JavaScript, JScript, VBScript, ...) werden vom Server unverändert an den Client gesendet und im Kontext des Browsers ausgeführt. Der Browser muß natürlich dazu in der Lage sein, indem er die entsprechende Version der verwendeten Skriptsprache unterstützt. Das kann natuerlich zu Problemen fuehren. Der Quell-Code von client-seitigen Skripten ist auch für jedermann sichtbar, was ebenfalls von Nachteil sein kann.

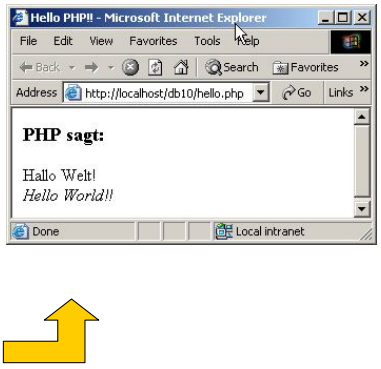
Dagegen werden server-seitige Skripte (Sprachen: PHP, JSP, ASP, ASP.NET, Perl, Python, ...) auf dem Server ausgeführt, bevor die Seite an den Client gesendet wird. Der Client bekommt also nur noch den vom Skript generierten HTML-Code zu sehen, nicht mehr den Quell-Code des Skriptes selbst. Der Client (Browser) muß auch keine kompatibilitätsbedingungen erfüllen, da beim Client ja nur HTML ankommt.

PHP Datei

- **PHP-Code wird in HTML eingebunden**
- **Markierung: <?php ... ?>**
- **Datei-Endung: .php**

hello.php

```
<html>
<head>
<title>Hello PHP!!</title>
</head>
<body>
<h3>PHP sagt:</h3>
<?php
    # Ausgabe mit echo und print()
    echo "Hallo Welt!<br/>";
    print("<i>Hello World!!</i>");
?>
</body>
</html>
```



Datenbanken I | 5 | SS 05, 17.06.2005 | FH Darmstadt, FB Informatik, Dr. P. Nevermann

Wie schon gesagt, wird PHP-Code in HTML-Seiten eingebunden. Zur Markierung des Codes dienen **<?php** für den Beginn und **?>** für das Ende. Alternativ funktioniert auch **<?** ... **?>** oder **<script language="php">** ... **</script>**.

Eine HTML-Seite welche PHP-Code enthält muß nun server-seitig in einer Datei mit der Endung **.php** (also nicht **.html** !!) gespeichert werden, damit der darin befindliche Script-Code zur Ausführung kommt. Alternativ gehen auch die Endungen **.phtml**, **.php3** und **.php4**. In XAMPP werden PHP-Dateien (wie HTML-Dateien) per Standardeinstellung unterhalb des Verzeichnisses *xampp/htdocs* gespeichert (die Einstellung kann natürlich auch geändert werden!).

Beispiel:

Das Beispiel auf der Folie zeigt die PHP-Datei *hello.php* welche im Verzeichnis *xampp/htdocs/db10* gespeichert ist (*db10* steht für "Datenbanken - 10. Vorlesung"). Im Browser kann man dann diese Seite unter der Adresse *http://localhost/db10/hello.php* testen.

Die Funktion *print()*, die in unserem Beispiel verwendet wurde, dient zur Ausgabe von Text, welcher typischerweise HTML-Tags enthalten kann. Letztendlich wird in einer PHP-Datei bei der server-seitigen Ausführung eine PHP-Code-Sequenz <?php ... ?> durch das Ergebnis der enthaltenen Aufrufe der *print()* Funktion ersetzt.

Eine Alternative zur print() Funktion für die Ausgabe ist mit der echo-Anweisung:

```
<?php
   echo "Hello World!!";
?>
```

Achtung:

Das Aurgument der print() Funktion ist i.d.R. eine Zeichenkette die mit doppelten Hochkomma (**"**) eingeschlossen wird. Da die Zeichenkette selbst wieder Hochkomma enthalten kann (z.B. für Attributwerte von HTML-Tags) muß man u.U. innerhalb des Wertes mit einfachen Hochkomma (**'**) arbeiten. Beispiel:

```
print("<hr size='20'/>");
```

Alternativ koennen auch Escape-Sequenzen verwendet werden, z.B. \" fuer die inneren Hochkommas.

Formal gesehen weist die PHP-Syntax eine Reihe von Ähnlichkeiten mit bekannten Programmiersprachen (Java, C, C++) auf, was das Erlernen der Sprache sehr vereinfacht:

•PHP-Anweisungen enden immer mit '**;**' (Semicolon)

•Kommentar-Zeilen können mit "//" oder '**#**' eingeleitet werden

•Kommentare können mit "/**" und "**/" eingeschlossen werden

Die Funktion print() ist eine von über 1000 verfügbaren Funktionen aus der umfangreichen PHP-Bibliothek. Dabei werden folgende Aufgabenbereiche abgedeckt:

•String-Verarbeitung (die meisten davon beginnen mit dem Präfix *str_*)

•Datum+Zeit-Verarbeitung (z.B. die Funktion *date()*)

•Zugriff auf das Dateisystem (*fopen()*, *fread()*, *fwrite()*, *fclose()*)

•Vesenden und empfangen von e-Mails

•Zugriff auf MySQL Datenbanken (Präfix *mysql_*).

Außerdem sind noch eine ganze Reihe weiterer PHP-Bibliotheken erhältlich, welche noch Zugriff auf andere Datenbanken als MySQL erlauben (z.B. Oracle), Verarbeitung von ZIP-Archiven, XML Verarbeitung, mathematische Funktionen, etc.

PHP Datentypen, Variablen und Operatoren

- **Datentypen**
  - Integer, Float, String, Boolean
  - Array, Object
- **Variablen**
  - beginnen mit $
  - typfrei
- **Operatoren**
  - Arithmetik
    - **+**, **-**, *****, **/**, **%** (modulo)
  - String Verkettung
    - **.**
  - Vergleiche
    - **==**, **===**, **!=**, **<**, **<=**, **>**, **>=**
  - Logisch
    - **!** (not), **&&**, **and**, **||**, **or**, **xor**

Datenbanken I | 7 | SS 05, 17.06.2005 | FH Darmstadt, FB Informatik, Dr. P. Nevermann

Variablen

Variablen sind in PHP typfrei und beginnen mit '$' (Dollar-Zeichen). Für den Variablennamen können Buchstaben, Ziffern und '_' (Unterstrich) verwendet werden. Dabei darf das erste Zeichen nach $ keine Ziffer sein. Also sind *$anzahl*, *$_preis2* gültige Variablennamen, *$2samkeit* aber nicht. Groß-/Kleinschreibung spielt in PHP eine Rolle, d.h. *$anzahl* und *$ANZAHL* sind unterschiedliche Variablen.

Im Unterschied zu den meisten Programmiersprachen sind PHP-Variablen nicht typ-behaftet (Zahl, String, ...) sondern können Werte beliebigen Typs aufnehmen. Außerdem muß eine PHP-Variable nicht explizit deklariert werden, sondern wird von PHP automatisch deklariert, sobald sie zum ersten Mal verwendet wird.

Datentypen

PHP kennt vier primitive Datentypen: Integer (Ganzzahl), Float (Gleitkommazahl), String (Zeichenkette) und Boolean (Boolescher Wert). Außerdem gibt es zwei zusammengesetzte Datentypen: Arrays und Objekte.

Integer

Integer entsprechen dem Datentyp long in C (d.h. -2^{31} bis $+2^{31} - 1$). Integer-Werte können decimal, hexadezimal oder octal angegeben werden:

•$i = 64;

•$j = 0xFF;

•$k = o44; // octal

Float

Floats entsprechen dem Datentype double in C und werden mit Decimalpunkt (z.B. 3.14159) oder in der wissenschaftlichen Notation (z.B. 1.07E+8) angegeben.

String

Strings werden durch einfache (') oder doppelte (") Hochkomma begrenzt (z.B. $s = "Hello World!" oder $txt = 'mitarbeiter'). Allerdings besteht ein wesentlicher Unterschied: Escape-Sequenzen und Variablen werden nur in Strings mit doppelten Hochkomma ausgewertet bzw. substituiert, wie das folgende Beispiel zeigt:

•Folgender PHP-Code (quotes.php):

```
<?php
$a = "Welt";
print("\x40 Hallo $a<br/>");
print('\x40 Hallo $a<br/>');
?>
```

produziert folgende Ausgabe (x40 ist die Hex-Darstellung des @-Zeichens):

```
@ Hallo Welt
\x40 Hallo $a
```

Praktisch ist auch die *heredoc-Syntax*, bei der doppelte Hochkomma keine Escape-Sequenz benötigen:

•Die Funktion nl2br wandelt noch \n (newline) in
 (HTML-Zeilenumbrüche) um (heredoc.php):

```
<?php
$a = <<<ENDE
Im Unterschied zu den meisten Programmiersprachen
sind PHP-Variablen nicht typisiert ("Zahl", "String", ...)
sondern können Werte beliebigen Typs aufnehmen.
ENDE;
print(nl2br($a));
?>
```

Boolean

Der boolsche Typ hat zwei Werte: true und false, z.B. $schalter = false; $first = true;

Arrays

Wie in Java oder C gibt es in PHP auch Arrays. Beispiel zur Verwendung (array.php):

```
•$a[0] = "Montag";
$a[1] = "Dienstag";
```

Folgende Zuweisung geht auch, wobei PHP automatisch den nächsten freien numerischen Index vergibt:

```
•$a[] = "Mittwoch";
$a[] = "Donnerstag";
```

Arrays können in PHP auch mit Strings indiziert werden (*assoziatives Array*):

```
•$a["Montag"] = "Monday";
$a["Dienstag"] = "Tuesday";
```

Einige Systemarrays (*siehe Bsp. systemvars.php*)

- •$GLOBALS[] Array der globalen Skript-Variablen (name->wert)
- •$_ENV[] Array der Umgebungsvariablen auf dem Server
- •$_SERVER[] Array mit webserver-bezogenen Variablen

Operatoren

Zur Bildung von Ausdrücken stehen in PHP eine ganze Reihe von Operatoren zur Verfügung, welche auch aus den Programmiersprachen (Java, C, C++) bekannt sind. Dazu gehören:

•Arithmetische Operatoren:
$a + 5
$b * 3.14159

•String-Verkettung:
"Der Typ ist ".gettype($a)

•Vergleiche:
$a == "Hello"
$b >= 64
$c != true

•Logische Verknüpfungen:
(gettype($a) == "integer") || ($b > 0 && $b < 10)
!($c == "yes")

Zudem sind verkürzende Schreibweisen möglich, wie wir sie auch aus Java oder C kennen:

•$i++	⇔	$i = $i + 1
•$i--	⇔	$i = $i – 1
•$a += 10	⇔	$a = $a + 10
•$s .= "Welt"	⇔	$s = $s . "Welt";

PHP Kontrollstrukturen

```
if (expr) {
  ...
} elseif (expr) {
  ...
} else {
  ...
}
```

```
while (expr) {
  ...
}
```

```
do (expr) {
  ...
} while (expr);
```

```
for (start_expr, cond_expr, iter_expr) {
  ...
}
```

```
switch (expr) {
  case val:
    ...
    break;
  default:
    ...
    break;
}
```

```
foreach (array as $value) {
  ...
}
```

```
foreach (array as $key=>$value) {
  ...
}
```

Datenbanken I | 10 | SS 05, 17.06.2005 | FH Darmstadt, FB Informatik, Dr. P. Nevermann

Die PHP-Syntax für Kontrollstrukturen ist sehr ähnlich mit der von Java, C oder C++.

Beispiele:

•if

```
if ($a > 0) {
   print("Das Ergebnis ist positiv");
} elseif ($a == 0) {
   print("Das Ergebnis ist null");
} else {
   print("Das Ergebnis ist negativ");
}
```

•switch

```
switch ($i) {
   case 0:
      print("null");
      break;
   case 1:
      print("eins");
      break;
   default:
      print("sonstwas");
}
```

•while

```
while ($i < 100) {
   print("Hier ist $i");
   i++;
}
```

•**for**

```
for ($i = 0; $i < 100; $i++) {
    print("Hier ist schon wieder $i");
}
```

•**foreach**

```
$a[] = "Null";
$a[] = "Eins";
$a[] = "Zwei";
print("<table border='1'>");
foreach($a as $value) {
  print("<tr><td>$value</td></tr>");
}
print("</table>");
```

•Nochmal **foreach**

```
$b["Zero"] = "Null";
$b["One"] = "Eins";
$b["Two"] = "Zwei";
print("<table border='1'>");
foreach($b as $key=>$value) {
  print("<tr><td>$key</td><td>$value</td></tr>");
}
print("</table>");
```

Schleifen abbrechen: break und continue

Schleifen (while, do ... while, for und foreach) können mit der Anweisungen *break* abgebrochen werden. Mit der Anweisung *continue* kann der Rest einer Iteration übersprungen werden, so daß mit der nächsten Iteration fortgefahren wird.

PHP Funktionen

```
printarray($_ENV);
print("Anzahl: ".arraysize($_ENV));
```

```
function printarray($array, $title='') {
    print("<hr size='5'/>$title<table border='2'>");
    foreach($array as $k=>$v) {
        print("<tr><td>$k</td><td>$v</td></tr>");
    }
    print("</table>");
}

function arraysize($array) {
    $size = 0;
    foreach($array as $v) {
        $size++;
    }
    return $size;
}
```


Funktionen werden in PHP mit dem Schlüsselwort *function* deklariert. Funktionen können optional Parameter haben und/oder einen Wert zurückgeben. Funktionswerte werden mit der *return* Anweisung zurückgegeben.

Funktionen können auch mit optionalen Parametern deklariert werden, indem ein Standard-Parameterwert angegeben wird. Beispiel:

•Deklaration einer Funktion mit optionalem Parameter:

```
function zufall($n=10) {
   return rand(0, $n);
}
```

•Funktionsaufrufe:

```
$z = zufall();      // liefert Zufallszahl zwischen 0 und 10
$z = zufall(20);  // liefert Zafallszahl zwischen 0 und 20
```

Funktionsargumente können als *Wert* oder als *Referenz* übergeben werden. In den vorangegangenen Beispielen wurden Argumente als *Wert* übergeben, d.h. Änderung des Parameterinhalts innerhalb der Funktion bleibt ohne Auswirkung auf das der Funktion übergebene Argument. Wird eine Argument als *Referenz* übergeben, trifft gerade das Gegenteil zu, d.h. Änderungen des Parameterinhalts innerhalb der Funktion werden außerhalb sichtbar. Bei dem als Referenz übergebene Argument muß es sich also immer um eine Variable handeln.

Um Argument als Referenz zu übergeben, gibt es in PHP eine besondere (an C angelehnte) Syntax mit vorangestelltem '**&**':

•Deklaration einer Funktion:

```
function verdoppel($N) {
   $N = $N * 2;
}
```

•Funktionsaufruf:

```
$k = 7;
verdoppel($k);
print($k);                    // liefert 7
verdoppel(&$k);
print($k);                    // liefert 14
```

Variablen innerhalb einer Funktion haben einen *lokalen* Geltungsbereich, d.h. sind außerhalb der Funktion nicht verfügbar. Globale Variablen sind dagegen auf höchster Ebene im Skript (d.h. außerhalb von Funktionen) allgemein verfügbar. Allerdings sind globale Variablen nicht unmittelbar innerhalb von Funtionen verfügbar, sondern müssen zuerst per *global* Anweisung bekanntgegeben werden. Alternativ kann innerhalb einer Funktion auch über $GLOBALS-Array auf globale Variablen zugegriffen werden.

•Beispiel:

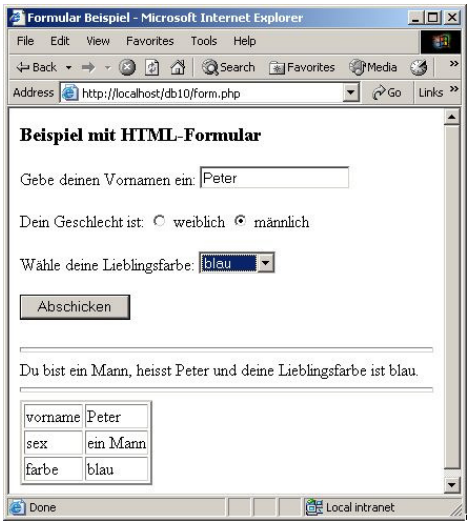
```
$a = "Hello World";
f1();
f2();
function f1() {
   echo "f1: ".$a;              // liefert nichts
   echo "f1: ".$GLOBALS['a'];   // liefert "Hello World"
}
function f2() {
   global $a;
   echo "f2: ".$a;              // liefert "Hello World"
}
```

Include-Dateien

Zwecks Wiederverwendung und besserer Übersichtlichkeit kann man PHP-Code (z.B. Funktionen, Konstanten, etc.) in sogenannte Include-Dateien auslagern. Diese haben die Endung **.inc** und werden im Skript mit der include() Funktion eingebunden:

```
include("myfunc.inc");
```

HTML Formulare



Zur Übergabe von Variablen aus einem HTML-Formular an das Skript stellt PHP eine Reihe von System-Arrays zur Verfügung (allesamt assoziative Arrays mit einer name->wert Zuordnung):

•$_GET
Variablen der GET Methode. Diese werden über die URL übergeben, z.B. http://localhost/db10/test.php?a=1&b=2 liefert $_GET['a'] = 1 und $_GET['b'] = 2.

•$_POST
Variablen der POST Methode, stammen typischerweise aus Formularen, bei denen die POST-Methode verwendet wird. Diese Daten werden nicht über die URL gesendet, sondern im Request-Body transportiert.

•$_COOKIE
Alle vom Client gesendeten Cookies.

•$_REQUEST
Zusammenfassung der $_GET, $_POST und $_COOKIE Arrays. Standardmäßig werden die Parameter in der Reihenfolge 'GPC' (d.h. GET->POST-COOKIE) geladen, wobei Parameter mit Namensübereinstimmung überschrieben werden.

Typischerweise wird man eine HTML-Seite mit dem Formular haben, z.B. myform.html. Im <form>-Element gibt man das PHP-Skript für die Verarbeitung der Formulardaten an und bestimmt noch die HTTP-Methode mit der das Formular gesendet werden soll (GET odet POST):

•<form action="myscript.php" method="post">

Innerhalb des <form>-Elements werden etliche Steuerelemente definiert sein, u.a.:

•Textfeld : <input type="text" name="parm1">

•Radio-Button: <input type="radio" name="parm2" value="Ja">

•Check-Box: <input type="checkbox" name="parm3" value="rot">

•Auswahl: <select single name="parm4"> mit <option>-Sub-Elementen

•Submit-Button: <input type="submit" value="OK">

Drückt der Client den Submit-Button, wird das Script, z.B. myscript.php, aufgerufen. Die Formulardaten holt sich das Skript aus $_GET, $_POST oder $_REQUEST, je nachdem mit welcher Methode das Formular gesendet wurde.

In dem Beispiel auf der Folie wurden Formular und Verarbeitungsskript in einer einzigen .php Datei zusammengefasst.

form.php

```
<html><head><title>Formular Beispiel</title></head>
<body>
<h3>Beispiel mit HTML-Formular</h3>
<?php
include("array.inc");
include("farben.inc");
?>
<form action="form.php" method="post">
  Gebe deinen Vornamen ein:
  <input type="text" name="vorname" value="*** dein Name ***"/><br/><br/>
  Dein Geschlecht ist:
  <input type="radio" name="sex" value="eine Frau" checked/> weiblich
  <input type="radio" name="sex" value="ein Mann"/> männlich<br/><br/>
  Wähle deine Lieblingsfarbe:
  <select single name="farbe">
    <?php
    foreach ($farben as $farbe) {
      print("<option>$farbe</option>");
    }
    ?>
  </select><br/><br/>
  <input type="submit" value="Abschicken"/>
</form>
<?php
```

form.php (Fortsetzung)

```
print("<hr size='5'/>Du bist ".$_REQUEST['sex'].
      ", heisst ".$_REQUEST['vorname'].
      " und deine Lieblingsfarbe ist ".$_REQUEST['farbe']."."");
printarray($_REQUEST);
?>
</body>
</html>
```

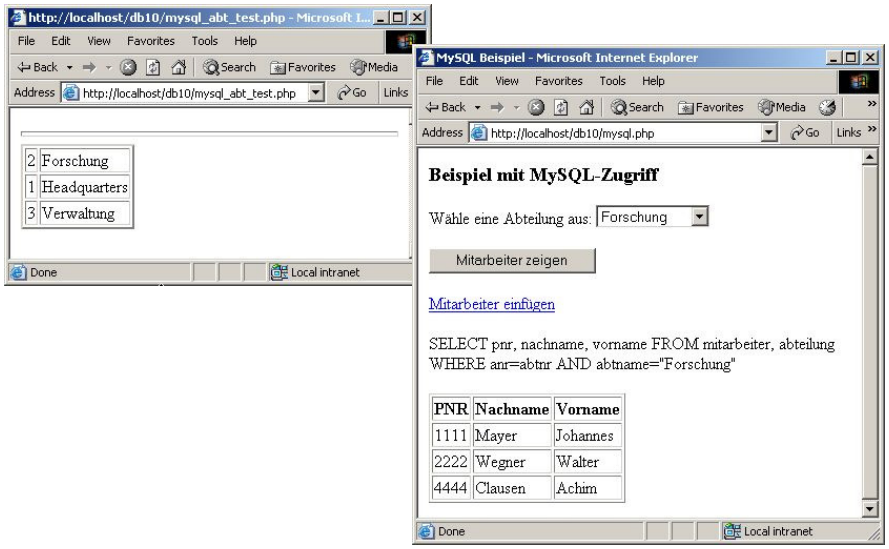
array.inc

```
<?php
function printarray($array, $title='') {
    print("<hr size='5'/>$title<table border='2'>");
    foreach($array as $k=>$v) {
        print("<tr><td>$k</td><td>$v</td></tr>");
    }
    print("</table>");
}
function arraysize($array) {
    $result = 0;
    foreach($array as $v) {
        $result++;
    }
    return $result;
}
?>
```

farben.inc

```
<?php
$farben[] = "rot";
$farben[] = "blau";
$farben[] = "gelb";
....
?>
```

MySQL



PHP stellt eine umfangreiche Bibliothek für den Zugriff auf MySQL bereit.

Der Rahmen für den Zugriff auf MySQL ist durch die beiden Funktionsaufrufe mysql_connect() und mysql_close() gegeben.

Die wichtigste mysql-Funktion ist mysql_query(), mit der beliebige Anfragen an MySQL gesendet werden können. Die Funktion mysql_query() gibt ein Ergebnis vom Typ "resource" zurück. Um daraus die Ergebnisdaten zu extrahieren, kann man die Funktion mysql_fetch_row() verwenden, welche ein Ergebnistupel als (numerisches) Array liefert.

Darüberhinaus gibt es eine ganze Reihe von mysql-Funktionen, welche vordefinierte MySQL-Operationen ausfuehren und eine bequemere Alternative zu manchem mysql_query() Aufruf darstellen soll:

- mysql_select_db("firma"); ⇔ mysql_query("use firma");
- mysql_create_db("cinecity"); ⇔ mysql_query("CREATE DATABASE cinecity");

Das folgende Beispiel zeigt die selbstgeschriebene Funktion getAbteilungen(), welche die die vorhandenen Abteilungen als assoziatives Array mit anr->abtname herausgibt. Die Funktion ist in der Include-Datei mysql_abt.inc definiert. Das Skript mysql_abt_test.php testet die Funktion (das Ergebnis ist auf der Folie oben links zu sehen):

mysql_abt_test.php:

```
<?php
include("array.inc");
include("mysql_abt.inc");
printarray(getAbteilungen());
?>
```

mysql_abt.inc:

```
<?php
// Gibt ein assoziatives Array anr->abtname zurück
function getAbteilungen() {
   $abteilungen = array();
   $con = mysql_connect("localhost", "root", "");
   mysql_select_db("firma");
   $query = "SELECT anr, abtname FROM abteilung";
   $result = mysql_query($query);
   while ($row = mysql_fetch_row($result)) {
      $abteilungen[$row[0]] = $row[1];
   }
   mysql_free_result($result);
   mysql_close($con);
   return $abteilungen;
}
?>
```

Das zweite Beispiel auf der Folie (unten rechts) verwendet die getAbteilungen() Funktion um die vorhandenen Abteilungen in einer Auswahl-Box eines Formulars zu präsentieren, und dann die Mitarbeiter der ausgewählten Abteiung anzuzeigen:

mysql.php:

```
<html><head><title>MySQL Beispiel</title></head>
<body>
<h3>Beispiel mit MySQL-Zugriff</h3>
<?php
include("array.inc");
include("mysql_abt.inc");
?>
<form action="mysql.php" method="post">
  Wähle eine Abteilung aus:
  <select single name="abteilung">
    <?php
    foreach (getAbteilungen() as $abtname) {
      print("<option");
      if ($abtname == $_REQUEST['abteilung']) {
        print(" selected");
      }
      print(">$abtname</option>");
    }
    ?>
  </select><br/><br/>
  <input type="submit" value="Mitarbeiter zeigen"/>
</form>
<?php
$con = mysql_connect("localhost", "root", "");
// mysql_select_db("firma");
mysql_query("use firma");
$abtname = $_REQUEST['abteilung'];
$query = <<<Q_END
SELECT pnr, nachname, vorname FROM mitarbeiter, abteilung
WHERE anr=abtnr AND abtname="$abtname"
Q_END;
echo $query."<br/><br/>";
$result = mysql_query($query);
print("<table border='1'>");
print("<tr><th>PNR</th><th>Nachname</th><th>Vorname</th></tr>");
while ($row = mysql_fetch_row($result)) {
    print("<tr><td>$row[0]</td><td>$row[1]</td><td>$row[2]</td></tr>");
}
print("</table>");
mysql_free_result($result);
mysql_close($con);
?>
</body>
</html>
```

Eine nützliche Alternative zu mysql_fetch_row() ist noch die Funktion mysql_fetch_assoc(), welche ein Ergebnistupel als assoziatives Array liefert. So kann man die Verarbeitung des Abfrage-Ergebnisses im obigen Skript auch folgendermaßen schreiben:

```
while ($row = mysql_fetch_assoc($result)) {
   print("<tr><td>".$row['pnr'].
         "</td><td>".$row['nachname'].
         "</td><td>".$row['vorname']."</td></tr>");
}
```

Mit diesem kurzen Überblick sollte gezeigt werden, daß es einfach ist mit PHP Web-Anwendungen zu schreiben und Web-Datenbanken einzubinden. Eine ganze Reihe von Themen konnten nicht angesprochen werden, weil es den Rahmen dieser Vorlesung sprengen würde:

•Fehlerbehandlung

•Zugriff auf das Dateisystem

•HTTP Sessions

•Einbindung von E-Mail Funktionen

•XML-Verarbeitung

•PDF-Verarbeitung

Literatur

Es gibt eine große Auswahl von Büchern über PHP – auch speziell mit Blick auf MySQL. Praktisch und handlich ist das O'Reilly Taschenbuch "PHP – kurz & gut" vom PHP-Erfinder Rasmus Lerdorf persönlich (O'Reilly, 2003).

Ebenfalls für diese Vorlesung habe ich das Taschenbuch "PHP – Webseiten dynamisch programmieren" von Michael Seeboerger-Weichselbaum (rororo, 2004) verwendet. Es hat mir den Einstieg erleichtert (ich hatte zunächst keine Ahnung von PHP). Viele Dinge blieben aber nach dem Lesen undeutlich und wurden erst nach Hinzunahme weiterer Texte klar.

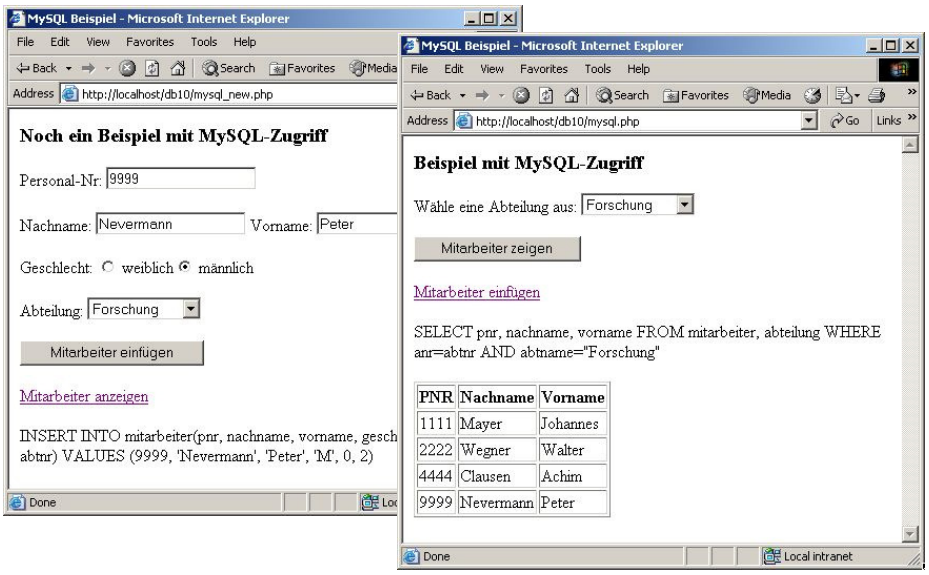
Links:

Offizielle PHP Dokumentation: http://www.php.net/docs.php

PHP-Builder Manual: http://www.phpbuilder.com/manual/

PHP Handbuch auf deutsch: http://www.phpreferenz.de/phpreferenz.php

MySQL



Zum Schluß noch ein Beispiel mit dem man neue Mitarbeiter einfügen kann.

Besonderes Augenmerk: das Formular "merkt" sich die Werte nach dem Abschicken der Formulardaten. Außerdem wird mit der Abfrage 'if ($_SERVER["REQUEST_METHOD"] == "POST")' zwischen dem ersten Aufruf der Seite (Methode = GET) und dem Schritt zur Verarbeitung der Formulardaten (Methode = POST) unterschieden. Schließlich werden noch mit der Funktion mysql_error() Fehlermeldungen der Datenbank angezeigt (z.B. "Duplicate entry '1122' for key 1", d.h. Personal-Nr 1122 schon vorhanden).

mysql_new.php:

```
<?php
include("array.inc");
include("mysql_abt.inc");
?>
<html><head><title>MySQL Beispiel</title></head>
<body>
<h3>Noch ein Beispiel mit MySQL-Zugriff</h3>
<form action="mysql_new.php" method="post">
 Personal-Nr:
 <input type="text" name="pnr" value="<?echo $_REQUEST['pnr']?>"/><br/><br/>
 Nachname:
 <input type="text" name="nachname" value="<?echo $_REQUEST['nachname']?>"/>
 Vorname:
 <input type="text" name="vorname" value="<?echo $_REQUEST['vorname']?>"/>
 <br/><br/>
 Geschlecht:
 <?php
  echo "<input type='radio' name='geschlecht' value='W'";
  if ($_REQUEST['geschlecht'] == "W") {
    echo " checked";
  }
  echo "/> weiblich";
  echo "<input type='radio' name='geschlecht' value='M'";
  if ($_REQUEST['geschlecht'] == "M") {
    echo " checked";
  }
  echo "/> männlich<br/><br/>";
 ?>
 Abteilung:
 <select single name="abteilung">
  <?php
  foreach (getAbteilungen() as $anr=>$abtname) {
    print("<option");
    if ($abtname == $_REQUEST['abteilung']) {
      $abtnr = $anr;
      print(" selected");
    }
    print(">$abtname</option>");
  }
  ?>
 </select><br/><br/>
 <input type="submit" value="Mitarbeiter einfügen"/><br/><br/>
 <input type="hidden" name="abtnr" value="<?echo $abtnr?>/>"
</form>
<a href="mysql.php">Mitarbeiter anzeigen</a><br/><br/>
```

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $con = mysql_connect("localhost", "root", "");
    mysql_select_db("firma");
    $pnr = $_REQUEST['pnr'];
    $nachname = $_REQUEST['nachname'];
    $vorname = $_REQUEST['vorname'];
    $geschlecht = $_REQUEST['geschlecht'];
    $abtname = $_REQUEST['abteilung'];
    $abtname = $_REQUEST['abtnr'];
    $query = "INSERT INTO mitarbeiter(pnr, nachname, vorname, geschlecht, gehalt, abtnr)
".
            "VALUES ($pnr, '$nachname', '$vorname', '$geschlecht', 0, $abtnr)";
    echo $query."<br/><br/>";
    mysql_query($query);
    echo mysql_error()."<br/><br/>";
    mysql_close($con);
}
?>
</body>
</html>
```

Sie können den Beispiel-PHP-Code dieser Vorlesung unter der Adresse:
http://www.fbi.fh-darmstadt.de/~pnevermann/ss05/Download/V10.ZIP
herunterladen.

Ausblick

- **Transaktionen** *(transaction)*
- **Konkurrenzsteuerung** *(concurrency control)*
- **Wiederanlauf** *(recovery)*