Veranstaltung
10033013

Systementwicklung

PHP – Teil 2

Uwe H. Suhl und Chris Bizer

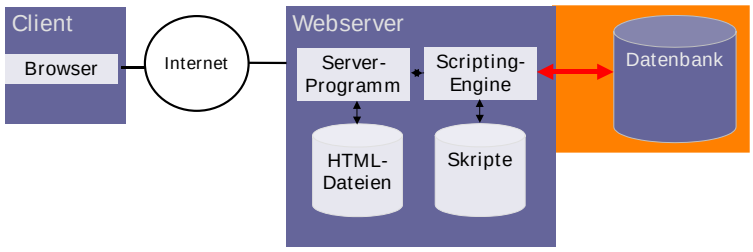
SS 2008

Gliederung

1. Datenbankanbindung
   1. Allgemeines zur Datenbankanbindung mit PHP
   2. Zugriff auf eine MS Access-Datenbank über ODBC
2. Objektorientierte Programmierung mit PHP
   1. Objekte mit PHP definieren
   2. Objekte in PHP nutzen
3. Sonstige nützliche PHP-Funktionen
   1. Zeichenkettenoperationen
   2. E-Mails verschicken mit PHP
   3. Zeit- und Datumsfunktionen

1. Datenbankanbindung mit PHP

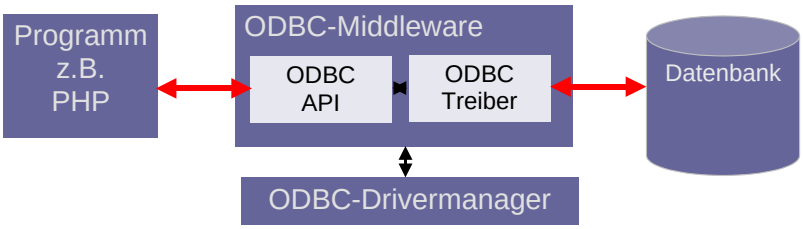
Datenbankanbindung mit PHP



- **PHP verfügt über datenbankspezifische Schnittstellen zu:**
  - **MySQL, MS SQL-Server, Oracle, Sybase, dBase, Informix**
  - **InterBase, mSQL, PostgreSQL, DBM**
- **Datenbankzugriffe in PHP erfolgen über:**
  - **datenbankspezifische Schnittstellen (hohe Performance)**
  - **Connectivity-Middleware wie ODBC (geringere Performance, aber OK)**
  - **Database-Abstraction-Layers wie adodb (http://adodb.sourceforge.net/)**
  - **PHP 5 verfügt über die eingebaute Datenbank SQLite**
- **Wir benutzen in der Übung eine Access-Datenbank, die wir über ODBC ansprechen.**

Open Database Connectivity (ODBC)

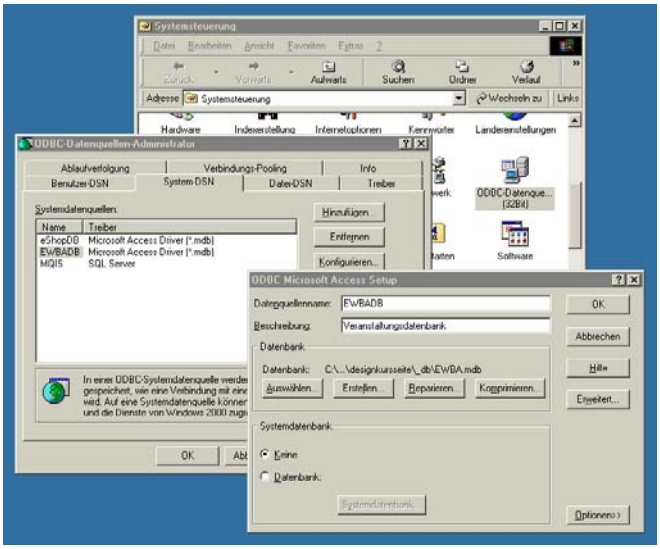
- ODBC ist eine Middleware-Komponente, die den Zugriff auf unterschiedliche Datenbanken über eine einheitliche Schnittstelle (API) erlaubt.



- Der ODBC-Standard wurde von Microsoft eingeführt.
- Es gibt ODBC-Treiber für nahezu jede Datenbank.
- Vorteil: Datenbanken sind austauschbar, ohne die Applikation umschreiben zu müssen (falls Standard-SQL verwendet, ANSI 89).
- Nachteil: Performance geringer als bei datenbank-spezifischen APIs.

Einrichten einer ODBC Datenquelle

- Mit dem Drivermanager „System Datasource Name“ (DSN) festlegen und mit der Datenbank verknüpfen.
- Die Datenquelle wird anschließend in PHP über diesen DSN angesprochen.



Beispiel 1: Staus aus der Datenbank holen

- Relationstyp stau_meldungen in der Datenbank staudb.mdb:

  stau_meldungen (id, datum, aktiv, autobahn, info)

```
// ODBC-Verbindung zur Datenbank aufbauen
$Connection_ID = odbc_connect("staudb", "", "");

// SQL Query-String definieren
$Query_String = "SELECT * FROM stau_meldungen
                 WHERE aktiv = '1'
                 ORDER BY autobahn, datum;";

// Abfrage ausfuehren
$Query_ID = odbc_exec($Connection_ID, $Query_String);
```


Beispiel 1: Ausgabe der Informationen

- Über eine While-Schleife werden alle Tupel des Abfrageergebnisses durchlaufen und
- die Inhalte in den HTML-Code der Seite geschrieben.

```
<?
// Schleife ueber alle Tupel
while(odbc_fetch_row($Query_ID)) { ?>
   <p>
   Autobahn: <? echo odbc_result($Query_ID, "autobahn") ?><br>
   Meldung: <? echo odbc_result($Query_ID, "info") ?><br>
   Datum: <? echo odbc_result($Query_ID, "datum") ?>
   </p>
   <hr size="2">
<? } ?>
```

ODBC-Befehle in PHP

Befehl	Bedeutung
`$Connection_ID = odbc_connect(„staudb", "", "");`	Baut eine Verbindung zur ODBC-Datenquelle staudb auf und liefert eine Connection_ID zurück. Bei Problemen wird FALSE zurückgegeben. Optional kann ein Benutzername und ein Passwort angegeben werden.
`$Query_ID = odbc_exec($Connection_ID, $Query_String);`	Führt das SQL-Kommando $Query_String über die Connection aus und liefert eine Query_ID zurück. Über diese Query_ID kann anschließend auf die Ergebnisse der Abfrage zugegriffen werden.
`odbc_fetch_row($Query_ID);`	Selektiert den nächsten Tupel der Abfrage $Query_ID
`odbc_fetch_row($Query_ID, 3);`	Selektiert den 3. Tupel der Abfrage $Query_ID
`echo odbc_result($Query_ID, „datum");`	Gibt Wert des Attributs "datum" des aktuellen Tupels der Abfrage $Query_ID aus.
`odbc_field_name($Query_ID,2)`	Liefert den Namen des Attributs Nummer 2
`odbc_field_type($Query_ID,2)`	Liefert den Datentyp des Attributs Nummer 2 (z.B CHAR 100)
`odbc_close($Connection_ID);` `odbc_free_result($Query_ID);`	Schließt Verbindung und gibt Speicher frei. Optional, da von PHP am Ende des Skripts automatisch ausgeführt.

Ablauf eines Datenbankzugriffs

Daten aus Datenbank lesen und in ein HTML-Template einfügen:

1. Verbindung zur ODBC-Datenquelle herstellen `odbc_connect()`
2. SQL-Kommando definieren `$Query_String =`
3. SQL-Kommando ausführen `odbc_exec()`
4. Ergebnisse der Abfrage ausgeben
   1. Tupel des Abfrageergebnisses auswählen `odbc_fetch_row()`
   2. Attributwerte in den HTML-Code ausgeben `echo odbc_result()`
5. ODBC-Verbindung wird von PHP beendet

Beispiel 2: Autobahn-Auswahl

```
$Connection_ID = odbc_connect(“staudb“, ““,““);

// Informationen zu den Autobahnen aus DB holen
$Query_String = "SELECT DISTINCT autobahn
                 FROM stau_meldungen
                 WHERE aktiv = '1'
                 ORDER BY autobahn;";
$Query_Menue = odbc_exec($Connection_ID, $Query_String);

// Ausgabe der Autobahninformationen
while(odbc_fetch_row($Query_Menue)) {
     echo "<p><a href='index.php?autobahn="
          . odbc_result($Query_Menue, 'autobahn')
          . "'>"
          . odbc_result($Query_Menue, 'autobahn')
          . "</a></p>";
};
```

- **Ergebnis:**

```
<p><a href='index.php?autobahn=A01'>A01</a></p>
<p><a href='index.php?autobahn=A07'>A07</a></p>
....
```


Situationsspezifische SQL-Befehle

- Die meisten Datenbankabfragen in Webapplikationen sind situationsspezifisch.
- Beispiel: Nur die Staus einer vom Nutzer gewählten Autobahn aus der Datenbank holen
- Vorgehen:
  - Dem Skript wird über einen Parameter das Auswahlkriterium übergeben
  - Ein passendes SQL-Kommando wird zusammengebaut und ausgeführt

```
// SQL-Query-String situationsspezifisch zusammenbauen
$Query_String = "SELECT * FROM stau_meldungen
                 WHERE aktiv = '1'
                 AND autobahn ='" . $_GET["autobahn"] . "'
                 ORDER BY datum;";
// SQL-Kommando ausfuehren
$Query_ID = odbc_exec($Connection_ID, $Query_String);
```

Formulareingaben in die Datenbank schreiben

- Nach ihrer Validierung werden die Formulareingaben über ein „INSERT INTO“Kommando in die Datenbank geschrieben:

```
// ODBC-Verbindung zur Datenbank aufbauen
$Connection_ID = odbc_connect(„staudb", "", "");

// SQL-Kommando zur Schreiben der Daten zusammenbauen
$Query_String = "INSERT INTO stau_meldungen
                     (autobahn, info, aktiv, datum)
                 VALUES
                     ('" . $_POST["autobahn"] . "','" .
                      $_POST["meldung"] .
                      "','1','" . $datum . "');";

// SQL-Kommando ausfuehren
$Query_ID = odbc_exec($Connection_ID, $Query_String);
```

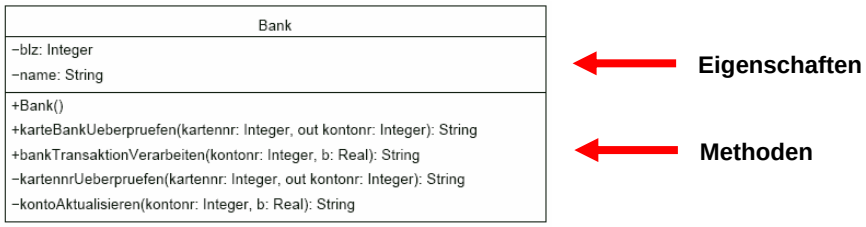
- Hinweis:
  - Beachte: Anführungszeichen im SQL-Kommando sind durch Hochkommata ‘ zu ersetzen.

2. Objektorientierte Programmierung mit PHP

Rückblick: Kapitel Entwurf

- Die Funktionalität des Systems wurde in Klassen aufgeteilt, die jeweils
  - Eigenschaften (Beschreibung eines Objekts) und
  - Methoden (Funktionalität eines Objekts) haben.

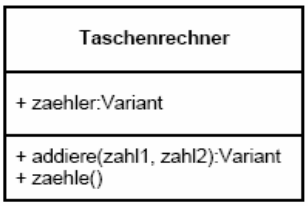
- Beispiel



Klassen, Methoden und Eingenschaften in PHP definieren

- Klassen werden in PHP mit dem Schlüsselwort class definiert.
- Eigenschaften mit dem Schlüsselwörtern public oder private.
- Methoden mit dem Schlüsselwort function().

```
// Klasse Rechner definieren
class Taschenrechner {
     // Eigenschaft: Zaehler
     public $zaehler = 0;
     // Methode: Addiere
     function addiere($zahl1, $zahl2) {
         $ergebnis = $zahl1 + $zahl2;
         return $ergebnis;
     }
     // Methode: Zaehle
     function zaehle() {
         $this->zaehler = $this->zaehler + 1;
     }
}
```



Parameter und Rückgabewerte

- **Methode mit 2 Parametern**

```
function addiere($zahl1, $zahl2) {
        $ergebnis = $zahl1 + $zahl1;
        return $ergebnis;
}
```

- **Methode ohne Rückgabewert**

```
function quadratAusgabe($zahl) {
        $zahl = $zahl * $zahl;
        echo „Das Quadrat ist “ . $zahl;
}
```

- **Methode mit Rückgabewert**

```
function quadrat($zahl) {
        $zahl = $zahl * $zahl;
        return $zahl;
}
```


Objekte in PHP benutzen

```
// Neues Objekt erzeugen
$rechner = new Taschenrechner();

// Eine Methode ohne Rückgabewert aufrufen
$rechner->zaehle();
$rechner->zaehle();

// Eine Methode mit Rückgabewert aufrufen
$summe = $rechner->addiere(3, 6);
echo $summe;

// Auf öffentliche Eigenschaften zugreifen
$stand = $rechner->zaehler;
echo $stand ;
```

$rechner ist eine Instanz der Klasse Taschenrechner

Um von außen auf eine Eigenschaft zugreifen zu können muss diese mit dem Schlüsselwort „public“ definiert sein.

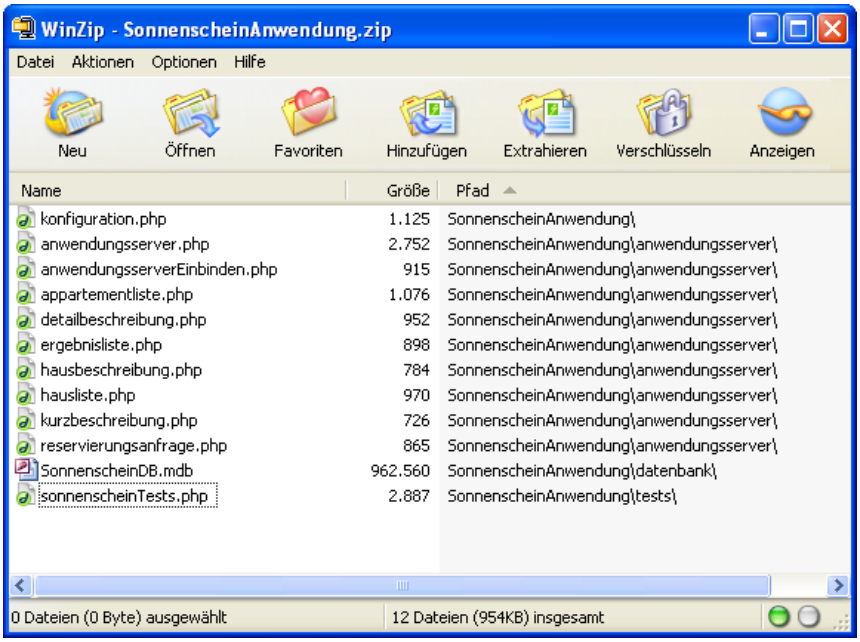
$this

Mit dem Schlüsselwort $this kann eine Methode einer Klasse auf Eigenschaften einer Instanz ihrer selbst zugreifen.

```
// Die Klasse Rechner definieren
class Taschenrechner {
     // Eigenschaft: Zaehler
     public $zaehler;

     // Methode: Zaehle
     function zaehle() {
         $this->zaehler = $this->zaehler + 1;
     }
}
// Die Klasse Rechner definieren
$rechner = new Taschenrechner();
$rechner->zaehle();
$rechner->zaehle();
echo $rechner->zaehler;
```


PHP Vorlage: SonnenscheinAnwendung.zip



3. Sonstige nützliche PHP-Funktionen

1. Zeichenkettenoperationen
2. E-Mails verschicken mit PHP
3. Zeit- und Datumsfunktionen

3.1 Strings teilen und zusammensetzen

Stringoperator	Erklärung
`$Teil = substr($String, $Start, $Laenge)`	Liefert einen Teil eines Strings
`$String = $Str1 . $Str2`	Verbindet Strings
`$Len = strlen($String)`	Bestimmt Anzahl der Zeichen eines Strings
`$Array = explode($Trennzeichen, $String)`	Zerteilt einen String anhand eines Trennzeichens und liefert ein Array mit den Teilen zurück.
`$String = implode($Trennzeichen, $Array)`	Verbindet alle Elemente eines Arrays zu einem String, separiert durch das Trennzeichen

```
$Str1 = "Hallo ich bin ein String";   // Zuweisung
echo substr($Str1, 6, 11);            // Ausgabe: ich bin ein
echo strlen($Str1);                   // Ausgabe: 24
$Array1 = explode(" ", $Str1);        // Erzeugt Array mit den Wörtern
echo $Array1[2];                      // Ausgabe: bin
$Str1 = implode(";", $Array1);        // Verbindet Elemente separiert durch ;
echo $Str1;                           // Ausgabe: Hallo;ich;bin;ein;String
```

Suchen und Ersetzen in Strings

Stringoperator	Erklärung
`$Pos = strpos($Heuhaufen, $Nadel [, $Start]);`	Liefert die Position des Suchstrings in der Zeichenkette oder FALSE, falls nicht gefunden. Optional kann eine Startposition angegeben werden
`$Str = str_replace($Nadel, $Neu, $Heuhaufen)`	Ersetzt alle Vorkommen von $Nadel in $Heuhaufen durch $Neu

```
$Str1 = "Hallo ich bin ein String";
echo strpos($Str1, "bin");            // Ausgabe:10
echo str_replace("in", "IN", $Str1); // Ausgabe:Hallo ich bIN eIN StrINg
```

- **Exkurs: Regular Expressions**
  - Mittels Regular Expressions lassen sich komplexe Suchmuster definieren.
  - Beispiel: Setze alle Wörter, die mit dem Buchstaben A beginnen und nicht am Anfang eines Satzes stehen, in Anführungszeichen.

Transformationen

Stringoperator	Erklärung
`$Str = chr($ASCII)`	Liefert ASCII -Zeichen z.B chr(10) für Zeilenumbruch im Quelltext
`$Str = urlencode($Str)`	kodiert Zeichenkette zur Weitergabe als URL-Parameter
`$Str = urldecode($Str)`	Decodiert URL-Zeichenkette
`$String = nl2br($String)`	Wandelt Zeilenumbrüche in -Tags um

```
$Str = "Beispielstring zum 'Ändern'";
echo urlencode($Str);          // Ausgabe: Beispielstring+zum+%27%C4ndern%27
$Str = „Erste Zeile" . chr(10) . "Zweite Zeile";
echo $Str;                     // Ausgabe: Erste Zeile
                                           Zweite Zeile
echo nl2br($Str);              // Ausgabe: Erste Zeile<BR>Zweite Zeile
```

3.2 Emails verschicken mit PHP

- **Anwendungsszenarien**
  - Bestätigung einer Bestellung, einer Anmeldung
  - Hinweise an den Administrator wenig besuchter Sites
  - Verschickten von Newslettern, vergessener Kennwörter
  - Web-basierte eMail-Clients (GMX, Hotmail)
- **Email-Konfiguration in der php.ini**

SMTP = mail.zedat.fu-berlin.de	Festlegung des SMTP-Servers, zum Verschicken der Emails
sendmail_from = shop@server.de	Festlegung des Absenders der Emails

- **Der PHP Mail-Befehl**

```
mail("chris@bizer.de", "Mein Betreff", "Mein Inhalt \n Mein Inhalt2");
```

Adressat | Betreffzeile | Text der Mail – Umbruch mit \n oder chr(10)

Beispiel Email

1. Text mit Stringoperationen zusammensetzen
2. und verschicken

```
// Adressat festlegen
$adressat = “chris@bizer.de“;
// Text der E-Mail zusammensetzen
$text_e-mail = "Hallo " . $name . "," . chr(10) . "vielen Dank
für Deine Staumeldung.";
// E-Mail verschicken
mail($adressat, "Staumeldung", $text_e-mail);
```

3.3 Zeit- und Datumsfunktionen

Intern arbeitet PHP mit UNIX-Timestamps. Ein Timestamp entspricht den seit Beginn der UNIX-Epoche (Januar 1 1970 00:00:00 GMT) bis jetzt vergangenen Sekunden.

Befehl	Erklärung
`$stamp = time()`	Gegenwärtiger UNIX-Timestamp
`$a = mktime(hh, min, sek, MM, DD, YYYY)`	Gibt den UNIX Timestamp/Zeitstempel für eine Zeitangabe zurück
`$stamp = getlastmod()`	Liefert den Timestamp der letzten Aktualisierung einer Datei.
`$d = date($Format [, $timestamp])`	Formatiert Datums- und Zeitangaben (falls keine Zeitangabe wird aktuelle Zeit genommen).

Format	Erklärung
`j d`	Tag (ohne/mit 0)
`n m`	Monat (ohne/mit 0)
`Y y`	Jahr (2- oder 4-stellig)
`s`	Sekunden
`i`	Minuten
`H h`	Stunden (24h/12h Format)

Beispielformate:

```
m/d/y       Ausgabe: 06/04/2001
j.n.Y       Ausgabe: 6.4.01
j.n.Y H:i:s Ausgabe: 6.4.01 14:03:56
```