

# Datenbank basierte Web- Anwendungen mit PHP

Konrad Wulf

[wulf@hls.de](mailto:wulf@hls.de)

Softwaretechnology Group

<http://www.hls.de/organization/st>

High Performance Computing Centre  
Stuttgart (HLRS)

# Hinweis

- Diese Folien hier liegen im Original unter [http://programming.jacomac.de/PHP\\_kurs.ppt](http://programming.jacomac.de/PHP_kurs.ppt).

# Inhalt

- Einführung in PHP
  - Was ist und was kann PHP?
  - Basiskonstrukte in PHP
  - Parameterübergabe von Seite zu Seite (POST/GET)
  - Personalisierung durch sessions und cookies
  - Eigenes Schreiben von Funktionen
- Grundsätzliches zum Thema Datenbankdesign und –management
  - Relationale Datenbanken – was ist das und wozu?
  - Übersicht über SQL
  - Referentielle Integrität
  - Transaktionen

# Inhalt fortgesetzt

- Weiterführende Themen
  - prozeduraler vs. objektorientierter Programmierstil
  - Trennung von Darstellung und Prozesslogik
  - Datenbankabstraktionsschicht und StandardDatenbanken
  - Formatüberprüfung von Daten (reguläre Ausdrücke etc.)
  - Authorisierung via HTTP vs. via PHP

# Was *ist* PHP?



- Eine serverseitige Skriptsprache zur Erstellung dynamischer Webinhalte. Das bedeutet:
  - wird zur Laufzeit übersetzt und ausgeführt (Interpretersprache)
  - wird auf der ausliefernden Maschine abgearbeitet => wie bei Perl, ASP und JSPs/Servlets. Im Gegensatz dazu (clientseitige Ausführung): Applets und Javascript
  - dynamisch, weil Aufrufergebnis abhängig von Benutzereingaben und/oder anderer äusserer Einflüsse
  - läuft i.d.R. innerhalb eines Webservers, wie z.B. Apache, Internet Information Server etc. Entweder als CGI oder als ISAPI Modul

# Wo bekommt man PHP?

- Von <http://www.php.net/downloads.php>
- Dokumentation in allen erdenklichen Sprachen und diversen Formaten unter <http://www.php.net/docs.php>

# Was *kann* PHP (1) ?



- Klassische prozedurale Programmierung, kann aber auch objektorientiert eingesetzt werden.
- kann direkt in HTML Seiten reingeschrieben werden, muss aber nicht.
- Native Anbindung an eine Vielzahl von Datenbanken
- Personalisierung von Daten durch Sessions

# Was *kann* PHP (2) ?



- Datenformatüberprüfung durch reguläre Ausdrücke
- Eine Reihe von Protokollen: HTTP, FTP, SSL, SMTP, IMAP...
- Zugriff auf Betriebssystemfunktionen
- Generierung dynamischer Bilder
- Kryptographische Funktionen
- etc...

# Basiskonstrukte in PHP

- PHP Code wird in ein bestimmtes Tag eingebettet: `<? [anweisungen] ?>`. Jede Anweisung wird mit einem Semikolon (;) abgeschlossen. Zum Beispiel so:
  - `<? echo ( "Aus dem Auge, aus dem Sinn." ); ?>`
  - `<?= "Aus dem Auge, aus dem Sinn." ; ?>`

# Variablentypen in PHP

PHP unterstützt acht primitive Typen.

- Vier skalare Typen:
  - Boolean\*
  - Integer
  - Fließkomma-Zahl (float)
  - String / Zeichenkette
- Zwei zusammengesetzte Typen:
  - Array
  - Object
- Und zuletzt zwei spezielle Typen:
  - Resource
  - NULL\*

# Variablen in PHP

- Variablen werden in PHP dargestellt durch ein Dollar-Zeichen (\$) gefolgt vom Namen der Variablen. Bei Variablen-Namen wird zwischen Groß- und Kleinschreibung unterschieden (case-sensitive).
- Variablen-Namen werden in PHP nach den gleichen Regeln wie andere Bezeichner erstellt. Ein gültiger Variablen-Name beginnt mit einem Buchstaben oder einem Unterstrich ("\_"), gefolgt von einer beliebigen Anzahl von Buchstaben, Zahlen oder Unterstrichen. Ein Buchstabe entspricht a bis z bzw. A bis Z oder einem ASCII-Zeichen von 127 bis 255 (0x7f bis 0xff).

```
<?php $var = "Bob";  
$Var = "Joe";  
echo "$var, $Var"; // outputs "Bob, Joe"  
$4site = 'not yet'; // invalid; starts with a number  
$_4site = 'not yet'; // valid; starts with an  
underscore  
$täyte = 'mansikka'; // valid; 'ä' is ASCII 228. ?>
```



So lasset uns zur Tat schreiten ;-)

# Vorbereitung Arbeitsplatz



- Jeder installiere bitte die Entwicklungsumgebung PHPEdit v0.8.0.25 auf seinem Rechner. Dazu bitte:
  - Doppelklicken auf die Datei [\\dotnet\php\\_kurs\phpedit\\_0.8\PHPEditSetup\\_0.8.25.exe](http://dotnet/php_kurs/phpedit_0.8/PHPEditSetup_0.8.25.exe)
- Hinweis: Diese Freeware kann auch im Internet unter <http://www.phpedit.net/products/PHPEdit/> runtergeladen werden. Kommerzielle Alternative dazu: [PHPEd von NuSphere](#)

# Warum eine IDE?

- Texteditoren mit Syntax-highlighting gibt es viele für PHP, doch richtig Spass macht es erst wenn man:
  - kontextbezogen in die Dokumentation springen kann, wenn
  - die verfügbare Funktionen und deren Signatur automatisch angezeigt werden (Code completion), wenn
  - bereits Programmiersprache und Webserver konfiguriert und integriert sind, wenn
  - es eine Überblicksnavigation für vorhandene Funktionen und Klassenmethoden gibt und nicht zuletzt
  - ein Debugger integriert ist, bei dem man den Programmablauf zu Laufzeit detailliert mitverfolgen kann

# Kontrollstrukturen in PHP

- In PHP gibt es die gängigen Kontrollstrukturen:
  - `while (bedingung) { befehle }`
  - `if (bedingung) { befehle } else { andere_befehle }`
  - `for ($i=0;bedingung;$i++) { befehle }`
  - `break`, `continue` in Schleifen (`while/for`)

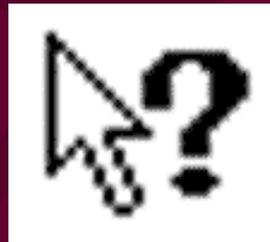
# Bsp. Auslesen e. Arrays

```
<? $voegel = array( "Kuckuck" , "Schneeeule" ,  
"Habicht" );  
  
while (list ($key, $val) = each ($voegel))  
{  
    echo $key." => ".$val."<br>\n";  
}  
?>
```

- Bitte dieses Skript zum Laufen bringen.
- Dazu den grünen Pfeil (  ) druecken.
- Die Ausgabe erscheint im unteren Fenster.

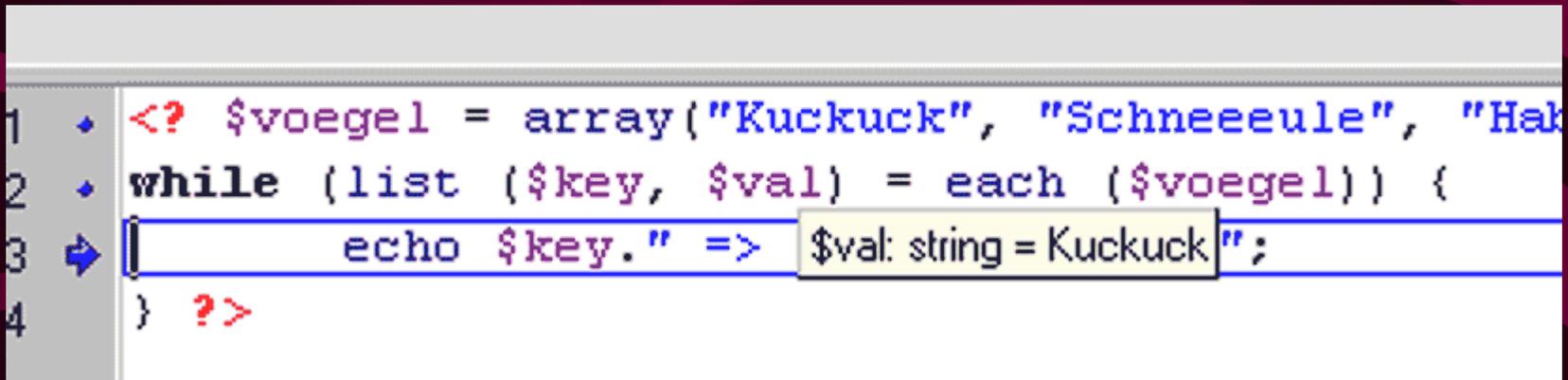
# Kontexthilfe in PHPEdit

- ...geschieht durch markieren eines Wortes und druecken von „Strg+F1“
- Im vorigen Beispiel dazu bitte mal das Wort „array“ markieren und die Kontexthilfe dazu aufrufen.



# Debuggen in PHPEdit

Zum Debuggen wieder grünen Pfeil drücken, danach aber diesmal auch „F8“ drücken. Ein blauer Pfeil durchläuft den Code Schritt für Schritt (s.Bild).



```
1 <? $voegel = array("Kuckuck", "Schneeeule", "Hak  
2 while (list ($key, $val) = each ($voegel)) {  
3     echo $key." => $val: string = Kuckuck";  
4 }
```

The screenshot shows a code editor window with a PHP script. The script is as follows:

```
1 <? $voegel = array("Kuckuck", "Schneeeule", "Hak  
2 while (list ($key, $val) = each ($voegel)) {  
3     echo $key." => $val: string = Kuckuck";  
4 }
```

The mouse cursor is hovering over the variable `$val` in the `echo` statement on line 3. A tooltip box is displayed, showing the current value of `$val`: `$val: string = Kuckuck`.

Wenn man mit dem Mauszeiger über eine Variable fährt, erfährt man den derzeitigen Wert von Ihr. In obigen Bild war der Mauszeiger gerade auf `$val`.

# Globale Variablen: GET

- array `$_GET`:
  - In diesem Array befinden sich alle an die Seiten übergebenen HTTP GET Parameter. Diese werden i.d.R. in durch den Aufruf einer jeweiligen URL angegeben => Alles was hinter dem Fragezeichen (?) steht, genannt **Querystring**.
  - Beispiel: <http://dotnet.rus.uni-stuttgart.de/php/kurs/beispiele/get.php?query=where+have+all+the+flowers+gone> => steht als `$_GET["query"]` dem Skript `get.php` zur Verfügung.
  - Hinweis: der Quellcode dazu liegt unter:  
[\\dotnet\kurs\beispiele\get.php](http://dotnet.kurs.beispiele.get.php)

# Übung zu GET

```
<html><TITLE>Beispiel GET</TITLE><BODY<?  
if (!empty($_GET["input"])) {  
    echo "input: " . $_GET["input"] ;  
}else{  
    echo "Du hast kein GET Parameter names input  
spezifiziert." ;  
} ?></BODY></html>
```

- Bitte obiges Beispiel zum laufen bringen.
- Jedem aus dem Schulungsraum steht ein Webspaces zur Verfügung unter [\\dotnet\russemX](http://dotnet.russemX), wobei X für Ihre Platznummer steht. Über den Browser lautet die Adresse dann <http://dotnet.rus.uni-stuttgart.de/php/russemX/>

# Übung zu GET (2)

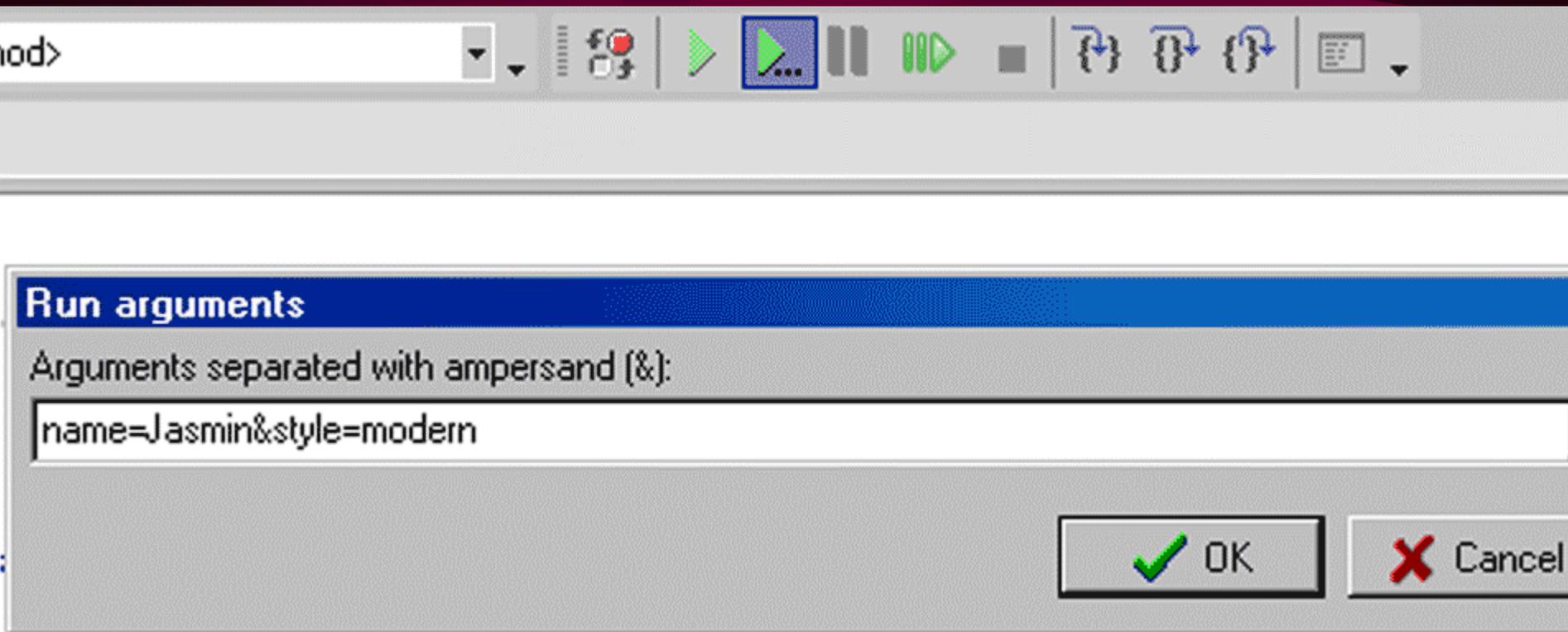
- Aufgabe:
  - Bitte **alle** Parameter auslesen, die per GET übergeben wurden.
- Beispiel-URL:  
<http://dotnet/php/kurs/beispiele/get.php?input=something&output=something+else>

# Lösung GET Aufgabe

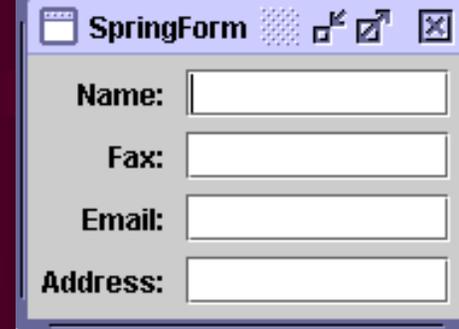
```
<HTML>
<HEAD>
<title>GET Array auslesen</title>
</HEAD>
<BODY>
<?
while (list ($key, $val) = each
    ($_GET)) {
    echo "$key => $val<br/>\n" ;
} ?>
</BODY>
</HTML>
```

# Debuggen mit GET Parametern

Um GET Parameter im Debugger mitzugeben, kann man den grünen Pfeil mit den DREI SCHWARZEN PUNKTEN benutzen.



# Globale Variablen: POST



SpringForm

Name:

Fax:

Email:

Address:

- Array `$_POST`:
  - Auf diesem Weg werden standardmässig Daten übergeben die mit einem HTML Formular erzeugt wurden.
  - Im Attribut „action“ vom jeweiligen `<form>` tag, wird das (PHP) Skript bestimmt, an das die POST Daten übermittelt werden sollen.

# Beispiel zu POST

## Schritt 1: das Formular erstellen

```
<html><body>
<form action="post_action.php" method="POST">
  Dein Name: <input type="text" name="name">
  Wie geht's?:
  <select name="how">
    <OPTION>gut</OPTION>
    <OPTION>k<b>ouml</b>nnte besser sein.</OPTION>
  </select><br><br>
  <input type="submit">
</form>
</body></html>
```

# Post\_action.php

2. Schritt: das empfangende Skript schreiben.

```
<HTML>
<HEAD><title>POST Beispiel</title></HEAD>
<BODY>
Hi <?= $_POST[ "name" ] ; ?> , <br>
Dein Zustand: <?php echo $_POST[ "how" ] ; ?> .
</BODY>
</HTML>
```

3. Schritt: ausführen unter <http://dotnet.rus.uni-stuttgart.de/php/russemX/>

# Globale Variablen: SESSION

- Ausgangsproblem:
  - Über HTTP werden keine ständigen Verbindungen zu den Servern hergestellt, sondern die Kommunikation läuft immer nach dem Schema „Anfrage“ => „Antwort“ (request/response). Deshalb können sich Webseiten per se nicht „erinnern“ (= stateless protocol).
- Lösung durch Sessions:
  - Entweder server- oder clientseitig (=> cookies) werden pro User-Agent (= Browser etc.) individuelle Parameter gespeichert, auf die dann alle PHP Skripte Zugriff haben. Diese Parameter müssen dann nicht mehr mit POST oder GET übermittelt werden. Lediglich eine Session ID muss von Seite zu Seite übermittelt werden.

# Beispiel zu SESSION

1. Anfangsseite: hier wird die Session in's Leben gerufen und im Link zur nächsten Seite wird die Session\_id als einziger Parameter weitergegeben.

```
<? session_start( ) ;  
$_SESSION[ "gaga" ] = "Diese Information  
sollte erhalten bleiben." ; ?>  
<HTML><BODY><br>  
<A href="session2.php?<?=SID?>" >weiter zu  
Seite 2</A>.  
</BODY></HTML>
```

# Beispiel zu SESSION (2)

2. Schritt: Alle Variablen aus dem Session Array werden ausgelesen.

```
<? session_start();?>
<HTML><BODY><?
while (list ($key, $val) = each ($_SESSION)) {
    echo "$key => $val<br/>\n";
} ?>
</BODY></HTML>
```

# Übung zu Session

- Voriges Beispiel bitte ausprobieren, also laufen lassen.
- Bitte einfach mehrere Parameter in die Session schreiben und auf nächster Seite auslesen.

# Schreiben eigener Funktionen

- Das Schreiben eigener Funktionen erleichtert die Wiederverwendung und Anpassung von Code.

```
<? function convertDM2Euro ($mark = 1)
{
    $euro = $mark * 1.95583 ;
    return $euro;
}

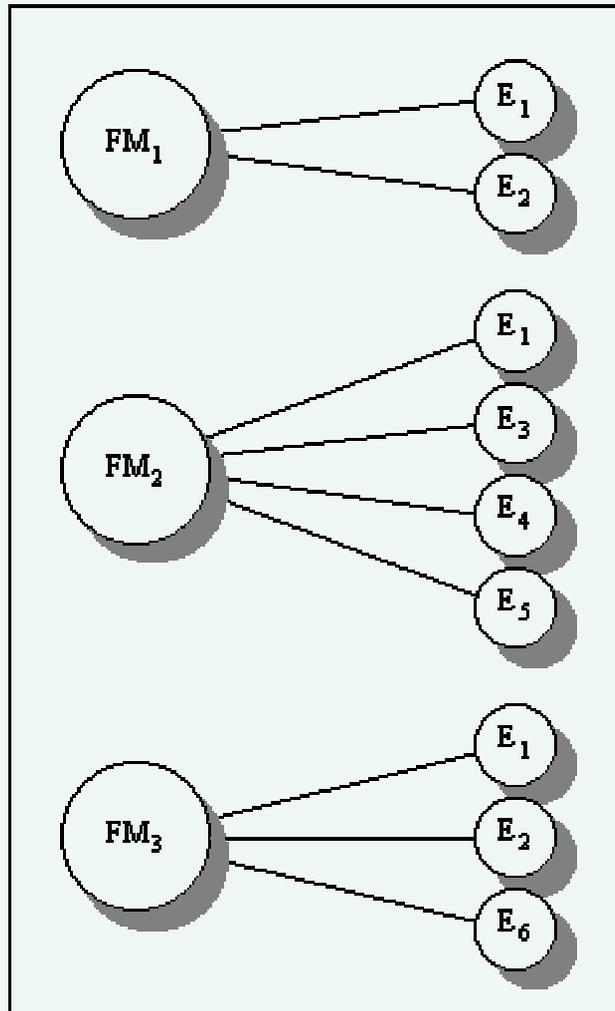
// hier der Aufruf mit Ausgabe
echo "Euro: " . convertDM2Euro( ) ; ?>
```

# Übung Functions

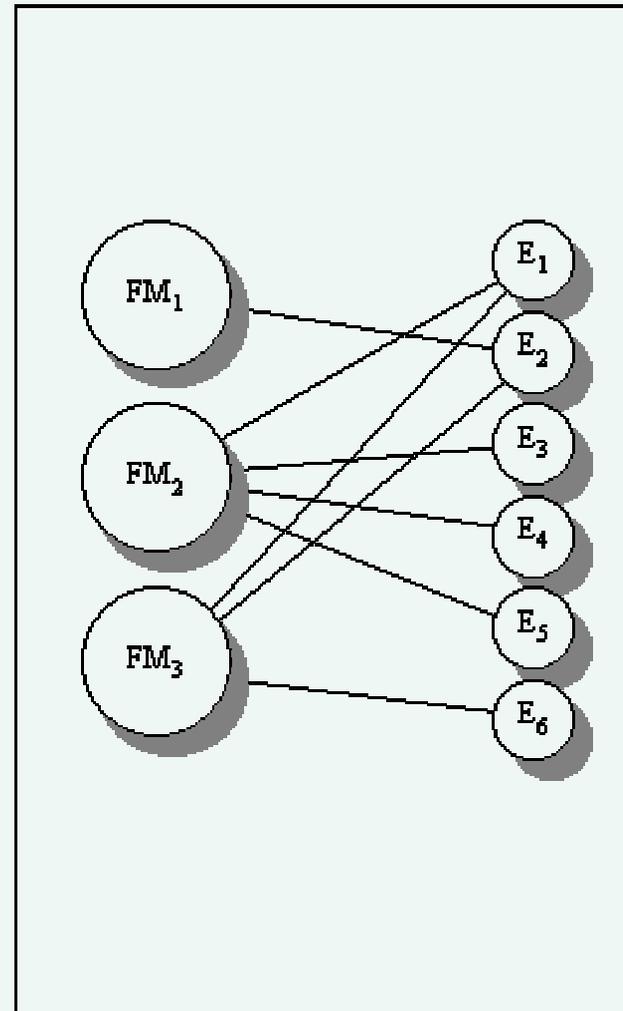
- Schreiben Sie eine Funktion `show_array($someArray)`, die ein übergebenes Array ausliest und anzeigt.

# Thema relationale Datenbanken

## One-to-Many Relationships

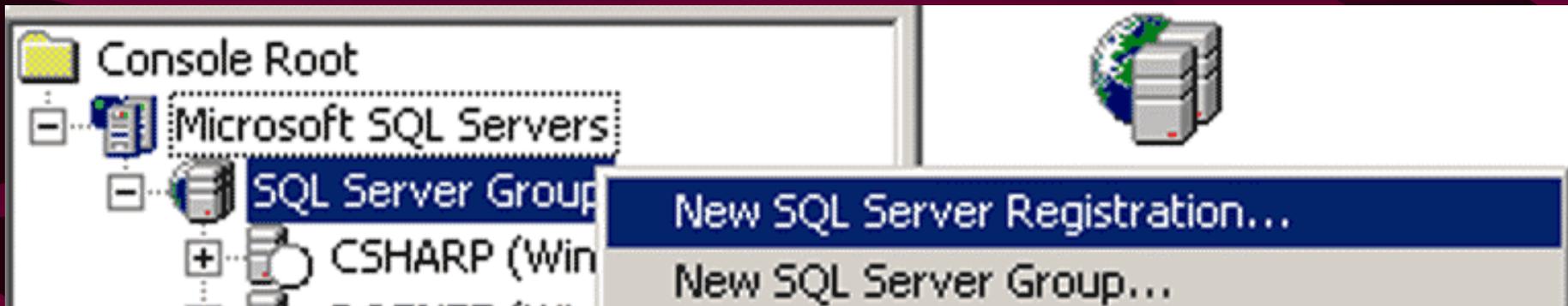


## Many-to-Many Relationship



# SETUP für die Benutzung von MS SQL

- „Start ▶ Programme ▶ Microsoft SQL Server ▶ Enterprise Manager“ bitte starten.
- Bei „SQL Server Gruppe“ per Kontextmenü „neue SQL Server Anmeldung“ wählen.



- Im jetzt erscheinenden Dialog bei Server „dotnet“ eingeben und „OK“ druecken.

# Benutzung MS SQL (2)

- Datenbanken-Knoten öffnen:
  - unter „RusKursXX“ steht jedem Teilnehmer eine eigene DB zur Verfügung.
  - Die DB „KursMaterial“ enthält Daten, auf die jeder Teilnehmer lesend zugreifen kann.
- Dokumentation fuer SQL Server liegt unter [\\dotnet\php\\_kurs\mssql7\mssql7.chm](\\dotnet\php_kurs\mssql7\mssql7.chm)

# Übersicht über SQL (1)

- Data **Definition** Language (DDL)
  - CREATE TABLE
  - ALTER TABLE
  - DROP TABLE
- Data **Manipulation** Language (DML)
  - INSERT INTO table VALUES (...)
  - DELETE FROM table
  - UPDATE table SET
- Reine **Abfragen**
  - SELECT something FROM table

# Beispiel Data Definition

```
CREATE TABLE wissens_quiz (  
  id int IDENTITY NOT NULL,  
  frage text NOT NULL,  
  antwort_a varchar(255) NOT NULL,  
  antwort_b varchar(255) NOT NULL,  
  antwort_c varchar(255) NOT NULL,  
  antwort_d varchar(255) NOT NULL,  
  richtige_antwort char(1) NOT NULL,  
  schwierigkeitsgrad int,  
  PRIMARY KEY (id)  
);
```

# Übung Data Definition

- Bitte zusätzlich den „Query Analyser“ aus gleicher Programmgruppe wie „Enterprise Manager“ aufrufen.
- Die „Create Table“-Anweisung von der vorigen Seite ausführen.
- Im Enterprise Manager kann nun die erstellte Tabelle gesehen werden.
- Bitte eine zweite Tabelle „Personen“ mit den Eigenschaften „Nachname“, „Vorname“ und „id“ erstellen.

# Foreign Key einführen (1)

- Wir wollen jetzt, dass jeder Eintrag in der „wissens\_quiz“-Tabelle genau einer Person zugeordnet ist, nämlich der, die die Frage eingestellt hat.
- Dazu brauchen wir eine weitere Spalte in `wissens_quiz` mit der `id` der jeweiligen Person. Das geht so:

```
ALTER TABLE wissens_quiz ADD person_id int  
NOT NULL
```

# Foreign Key einführen (2)

- Zur Überprüfung der Datenkonsistenz (gibt es die Person überhaupt?) bieten viele DB Server die Überprüfung dessen durch Anlegen von Fremdschlüsseln (Foreign Keys) an.

```
ALTER TABLE wissens_quiz ADD
```

```
CONSTRAINT FK_personen_quiz FOREIGN KEY
```

```
(person_id) REFERENCES Personen (id)
```



# Beispiel Data Manipulation

```
INSERT INTO wissens_quiz (frage, antwort_a,  
    antwort_b, antwort_c, antwort_d,  
    richtige_antwort, schwierigkeitsgrad)  
VALUES ('Wie heisst das beliebte  
    Mannschaftsspiel bei Harry Potter, das auf  
    Besen ausgetragen wird?', 'scotch', 'witch',  
    'quidditch', 'billitch', 'c', '8000');
```

- Bitte im Query Analyzer ausführen.
- Im Enterprise Manager bitte einen Eintrag fuer Personen hinzufügen.

# Übersicht über SQL (2)

- WHERE Einschränkungen
  - Gültig für reine Abfragen und DML, also:  
SELECT,INSERT,UPDATE,DELETE
- ORDER BY: Sortierung der Ergebnisse

# DML: UPDATE

```
UPDATE wissens_quiz  
SET    person_id = 1  
WHERE  id = 1
```

# Beispiel „reine Abfrage“

```
SELECT id, frage, antwort_a, antwort_b,  
       antwort_c, antwort_d, richtige_antwort,  
       schwierigkeitsgrad  
FROM wissens_quiz  
WHERE frage like '%harry%'
```

# PHP Beispiel SELECT

```
<?php
$host = "dotnet.rus.uni-stuttgart.de";
$user = "php_kurs";
$password = $user;
mssql_connect ($host, $user, $password);
mssql_select_db ("KursMaterial");
$query = "SELECT * FROM
    wissens_quiz_englisch";
$result = mssql_query($query);
while($row = mssql_fetch_array($result)) {
    echo $row["frage"] . "<br>\n";
} // while
?>
```

# Uebung SQL Abfrage in PHP

- Bitte ändern Sie das vorige Beispiel so ab, dass Sie statt auf „KursMaterial“ auf Ihre DB zugreifen (RusKursX).
- Statt auszulesen soll das Skript nun einen neuen Antrag generieren.

# Exkurs: Nützliche Funktionen

- Emails versenden

```
<? mail( "wulf@hlrs.de" , "betreff" ,  
"inhalt" , "From: wulf@jacomac.de" ); ?>
```

- Konfigurationsinformationen auslesen

```
<? phpinfo( ) ; ?>
```

- Fehleranzeige einstellen (0 = aus, 7 = alles anzeigen)

```
<? error_reporting( 0 ) ; ?>
```

# Übung „INSERT“

- Bitte fügen Sie in dem Beispiel unter [\\dotnet\kurs\workshop\\_registration](http://dotnet/kurs/workshop_registration) in der Seite „registration.php“ die fehlenden Befehle zum Speichern der Formulare Daten ein. Dazu bitte als erstes den gesamten Ordner in ein Verzeichnis auf dem eigenen Webservice kopieren. Die entsprechende Stelle, die ergänzt werden soll, ist mit einem mehrzeiligen PHP Kommentar (`/* */`) gekennzeichnet.

```
$query = "INSERT INTO Personen (Name,  
Einrichtung_Firma,Institut_Abteilung,  
Street_PostBox,"  
        ." ZipCode_Town, Country, Email, Telefon,  
Fax, damien_workshop, meta_workshop, Dinner)"  
        ." VALUES ('" .$_POST["Name"] ."',  
        ' " .$_POST["Einrichtung_Firma"] ."',      '"  
        .$_POST["Institut_Abteilung"] ."',  
        '" .$_POST["Street_PostBox"] ."',      '"  
        .$_POST["ZipCode_Town"] ."',  
        '" .$_POST["Country"] ."',      '"  
        .$_POST["Email"] ."',  
        '" .$_POST["Telefon"] ."',      '"  
        .$_POST["Fax"] ."',  
        '" .$_POST["damien_workshop"] ."',      '"  
        .$_POST["meta_workshop"] ."',  
        '" .$_POST["Dinner"] ."')";  
$result = MSSQL_QUERY($query);
```

# Lösung 2 „INSERT“

```
unset ( $_POST[ "Submit" ] );  
$spalten = implode( " , " ,  
    array_keys( $_POST ) );  
$eintraege = implode( "' , '" , $_POST );  
$query = "INSERT INTO personen  
    ( " . $spalten . " ) VALUES  
    ( ' " . $eintraege . " ' ) " ;  
mssql_query( $query ) ;
```

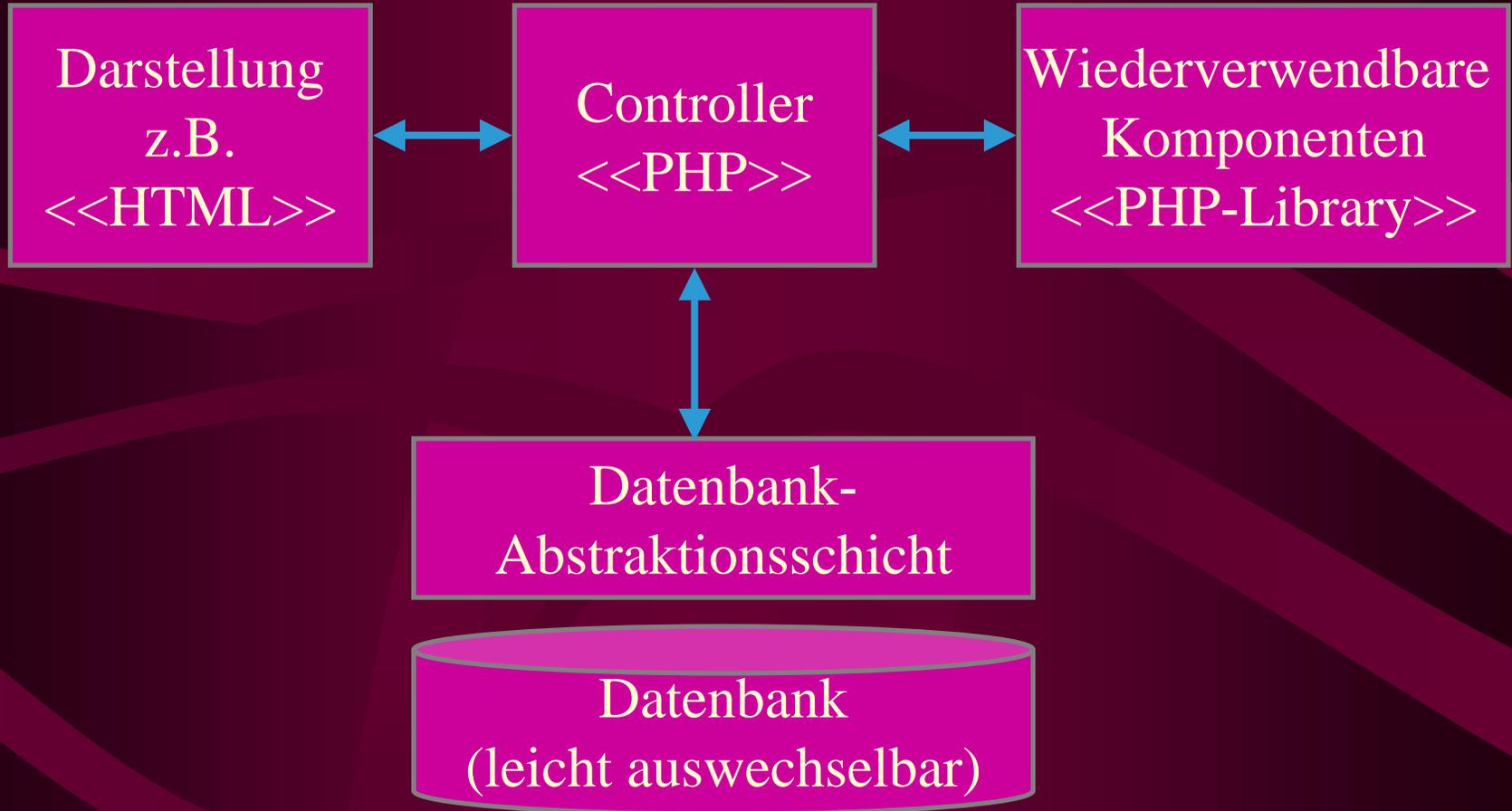
# Übung zu „SELECT“

- Bitte erstellen Sie auf Ihrem Webespace eine Kopie der Datei „tpl\_artikel.html“ unter [\\dotnet\kurs\shopping](#) und ersetzen die Platzhalter ##artikel\_name##, ##artikel\_preis## und ##artikel\_bild## mit Daten aus der Datenbank „KursMaterial“, Tabelle „nurtec\_preisliste“. Umbenennen in tpl\_artikel.php nicht vergessen!

# Lösung zu „SELECT“

```
<? mssql_connect( "dotnet.rus.uni-stuttgart.de" ,
    "php_kurs" , "php_kurs" );
mssql_select_db( "KursMaterial" );
$query = "SELECT ABBILDUNG, BEZEICHNUNG,
    PreisInEuro"
    . " FROM nurtec_preisliste"
    . " WHERE (id = 6)";
$result = mssql_query($query);
$row = mssql_fetch_array($result);?>
<html>
    ...<b>&nbsp;?<? = $row[ "BEZEICHNUNG" ]; ?></b>
    ..."
border="0" alt="Grossansicht">
    ...<b><? = $row[ "PreisInEuro" ]; ?> EUR</b>
inkl. MwSt.
...</html>
```

# Gutes Design



# Objektorientierter Programmierstil

```
<? class Utility {  
    var $debugInfo;  
    /* public_member methods */  
    function showArray($array, $indent = 0) {  
        echo "<xmp>";  
        $this->printArray($array, $indent);  
        echo "</xmp>";  
    }  
    [...]  
} ?>
```

Diese Utility-Datei liegt unter

<http://programming.jacomac.de/classLib/Utility.inc.php>

# OOP (2)

Diese Klasse „Utility.inc“ kann, um benutzt zu werden, wie folgt aufgerufen werden:

```
<?  
//einbinden  
include( "Utility.inc" );  
//instanzieren  
$util = new Utility();  
//methodenaufruf  
$util->showArray( $someArray );  
?>
```

Bitte ausprobieren.

# Übung OOP

- Schreiben Sie in PHP eine Klasse „Person.inc“, die die Methoden String getName() und void setName(String \$name) enthält.
- Schreiben Sie zudem eine zweite Datei „test\_person.php“, die diese Klasse testet.

# Lösung OOP: Person.inc

```
<?php
class Person {
    var $name;

    function getName() {
        return $this->name;
    }

    function setName($name_input) {
        $this->name = $name_input;
    }
} ?>
```

# Lösung OOP: test\_person.php

```
<?php
include ( "inc/Person.inc" );
$person = new Person( );
$person->setName( "Karl" );
echo "Die Person heisst:
    " . $person->getName( );
?>
```

# Ergänzung: Vererbung (1)

```
<?php
class Person extends Utility {
    [...]
    function setName($name_input) {
        $this->name = $name_input;
        if ($name_input == "Karl") {
            $this->debugInfo[] = "Schon wieder
ein Karl...";
        }
    }
}
}?>
```

# Ergänzung: Vererbung (2)

Nun können auch die geerbten Methoden und Variablen benutzt werden.

```
<?php
include ( "inc/Utility.inc" );
include ( "inc/Person.inc" );
$person = new Person();
$person->setName( "Karl" );
echo $person->showDebugInfo( );
?>
```

# Lesen von Dateien

Das untenstehende Skript liest die Datei

\\dotnet\kurs\wissens\_quiz\lib\wissens\_quiz\_daten.csv zeilenweise in ein Array und zerlegt die Spalten, die durch Tabulatoren (\t) getrennt sind, in ein Unterarray. Damit sind alle Daten fuer die Anwendung komfortatbel verfuegbar, ohne eine DB zu benoetigen.

```
<?php
$zeilen = file("wissens_quiz_daten.csv");
while(list($k, $zeile) = each ($zeilen)) {
    $zeilen[$k] = explode("\t", $zeile);
} // while
echo "frage: ".$zeilen[2][1];
?>
```

# Schreiben von Dateien

```
<?php
```

```
$neu_eintrag = "44\tWie alt muss man in  
Deutschland mindestens sein, um  
Bundespräsident werden zu  
können?\t18\t21\t35\t40\tD\t50000  
0\n";
```

```
$fp =  
fopen( "wissens_quiz_daten.csv" , "ab+" );  
fwrite( $fp, $neu_eintrag );  
fclose( $fp );
```

```
?>
```

# DatenbankAbstraktionsSchicht

- In einer Datenbankabstraktionsschicht werden alle Zugriffe auf die DB durch eine zentrale Datei mediatisiert.
- Sinn einer Datenbankabstraktionsschicht ist es, das Umstellen auf andere Datenbanken oder –quellen zu erleichtern. Auch können die Datenbankabfragen unabhängig vom restlichen Programmiercode entwickelt und getestet werden.

# Aufgabe zu DB Abstraktion

- Das Wissensquiz unter [\\dotnet\kurs\wissens\\_quiz](http://dotnet.rus.uni-stuttgart.de/php/kurs/wissens_quiz/) bzw. [http://dotnet.rus.uni-stuttgart.de/php/kurs/wissens\\_quiz/](http://dotnet.rus.uni-stuttgart.de/php/kurs/wissens_quiz/) zeigt drei entsprechende Implementierung für MS SQL, MySQL und SOAP (=> „inc/QuizDatabase.inc“). Die Implementierung in SOAP benutzt einen Webservice, der in Java implementiert ist, als Datenquelle.
- Bitte entwickeln Sie eine entsprechende Implementierung von QuizDatabase.inc, die statt auf eine DB zuzugreifen eine Datei benutzt.

# Lösung Datei-Version

Lösung zu finden unter

[http://programming.jacomac.de/php\\_kurs.zip](http://programming.jacomac.de/php_kurs.zip) ,

in `wissens_quiz/lib/QuizDatabase_file.inc`

# Benutzung der Template-Klasse

- Siehe Beispiel [\\dotnet\kurs\shopping\index.php](http://dotnet.kurs.shopping.index.php) bzw. <http://dotnet.rus.uni-stuttgart.de/php/kurs/shopping/>
- Hier wird zum einem die Darstellungsschicht durch Benutzung einer Template-Klasse getrennt und zum anderen die immer wiederkehrende Warenkorb-Funktionalität in eine Klasse gekapselt.
- Hinweis: Diese Klassen habe ich unter <http://programming.jacomac.de/> zur allgemeinen Verfügung gestellt.

```
<?  
$entries[ "artikel_name" ] =  
    $row[ "BEZEICHNUNG" ];  
$entries[ "artikel_bild" ] =  
    $row[ "ABBILDUNG" ];  
$entries[ "artikel_preis" ] =  
    $row[ "PreisInEuro" ];  
  
$tpl = new Template( "tpl_artikel.html" );  
$tpl->insertValues( $entries );  
echo $tpl->getText( );  
?>
```

- Das Template ist genau das gleiche wie in dem SELECT Beispiel zuvor. Platzhalter sind mit `##name##` gekennzeichnet. Näheres erfährt man, wenn man die `Template.inc` im `lib` Verzeichnis öffnet.

# Aufgabe Template

Bitte das shopping Verzeichnis unter [\\dotnet\kurs\shopping](http://dotnet.kurs.shopping) noch einmal neu in eigenes Verzeichnis kopieren. Dann bitte ein PHP skript mit Hilfe der Datei „lib/Template.inc“ schreiben, dass in der Datei „tpl\_artikel\_2.html“ die Platzhalter ##artikel\_name##, ##artikel\_preis## und ##artikel\_bild## ersetzt.

Die Datei Template.inc ist auch im web verfügbar unter <http://programming.jacomac.de/classLib/Template.inc.php> zur allgemeinen Weiterverwendung.

# Reguläre Ausdrücke

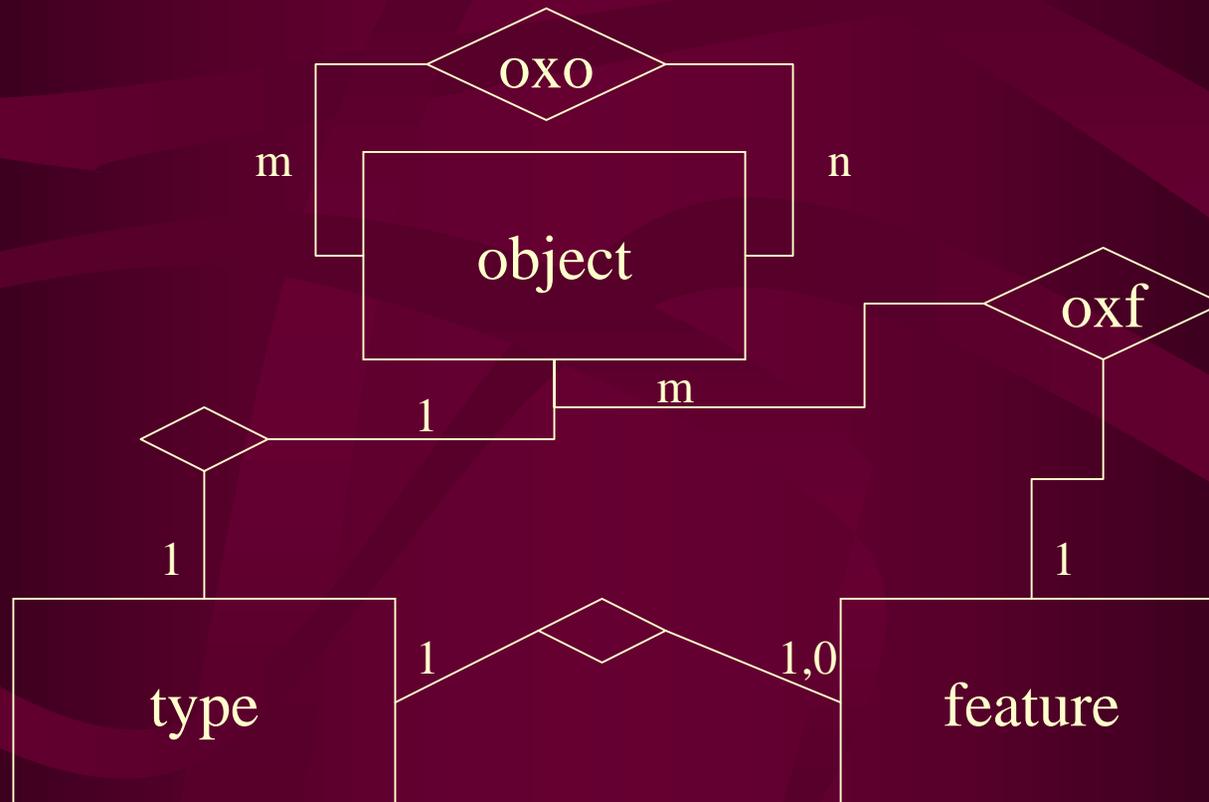
- ... dienen dem Auffinden von Mustern:
- Eckig Klammern ([ ]) definieren Wertebereiche für die jeweilige Position, z.B. [0-9] erkennt eine Zahl, [a-zA-Z] erkennt jeden Buchstaben, ausser Sonderzeichen+Umlaute
- {*min*, *max*} definiert die Anzahl der Wiederholungen, wobei die Angabe von *max* optional ist. z.B. f{2} findet „ff“ in „Stoff“
- Weiteres unter <http://etext.lib.virginia.edu/helpsheets/regex.html>

# Beispiel RegExp: Email-Format

```
<?
if (!empty($_POST["Email"]) &&
    !eregi("^([a-z0-9]+([\._-][a-z0-9]+)*@([a-z0-9]+([\._-][a-z0-9]+)*)+\.[a-z]{2,4})$", $_POST["Email"]))
{
    $errors[] = "The supplied email address
    does not have the right format.";
} ?>
```

# StandardDatenbank

- Jede Datenstruktur lässt sich in StandardDB nach z.B. unten stehendem Schema abbilden => Vorteil: DatenAbstraktionsSchicht muss nicht einmal angepasst werden.



# StandardDatenbank

- Die passende SQL DDL für MySQL liegt unter [http://programming.jacomac.de/classLib/setup/standardDB\\_mysql.sql](http://programming.jacomac.de/classLib/setup/standardDB_mysql.sql)

```
[...]  
# -----  
#  
# Table structure for table 'object'  
#  
  
CREATE TABLE object (  
  o_id int(11) NOT NULL auto_increment,  
  o_name varchar(255),  
  o_type_id int(11) NOT NULL,  
  o_text text,  
  o_sort varchar(255),  
  PRIMARY KEY (o_id),  
  UNIQUE o_id (o_id),  
  KEY o_id_2 (o_id)  
);  
[...]
```

# Std\_Datenbank\_Klasse

- Liegt unter  
\\dotnet\kurs\standard\_db\lib\StandardDB.in  
c

# Literatur-Empfehlung

Zwar nicht mehr das neueste, aber ein gutes Werk,  
sowohl als Einführung als auch als Referenz:

Castagnetto, J. et al. (2000). Professional PHP  
Programming. Wrox Press, Birmingham, UK.  
ISBN 1-861002-96-3

<http://isbn.nu/1861002963>