Grundlagen PHP

René Pfeiffer <pfeiffer@luchs.at>

CaT

23. April 2008

Inhaltsübersicht

Schwerpunkte PHP Einführung

- Ursprung von PHP
- PHP in HTML einbetten
- PHP Kontrollstrukturen & Variablen
- Überblick Funktionsumfang
- Verbinden mit Datenbanken
- Sicherheitsprobleme in PHP Skripten
- Code Beispiele

PHP - Geschichte

PHP - Geschichte

- PHP steht rekursiv für „PHP: Hypertext Preprocessor”

PHP - Geschichte

- PHP steht rekursiv für „PHP: Hypertext Preprocessor”
- 1995 Ursprung als PHP/FI „Personal Home Page / Forms Interpreter”

PHP - Geschichte

- PHP steht rekursiv für „PHP: Hypertext Preprocessor”
- 1995 Ursprung als PHP/FI „Personal Home Page / Forms Interpreter”
- 1997 PHP/FI 2.0

PHP - Geschichte

- PHP steht rekursiv für „PHP: Hypertext Preprocessor”
- 1995 Ursprung als PHP/FI „Personal Home Page / Forms Interpreter”
- 1997 PHP/FI 2.0
- 1997 komplettes Neuschreiben als PHP3

PHP - Geschichte

- PHP steht rekursiv für „PHP: Hypertext Preprocessor”
- 1995 Ursprung als PHP/FI „Personal Home Page / Forms Interpreter”
- 1997 PHP/FI 2.0
- 1997 komplettes Neuschreiben als PHP3
- 1999 komplettes Neuschreiben des Kerns als PHP4

PHP - Geschichte

- PHP steht rekursiv für „PHP: Hypertext Preprocessor”
- 1995 Ursprung als PHP/FI „Personal Home Page / Forms Interpreter”
- 1997 PHP/FI 2.0
- 1997 komplettes Neuschreiben als PHP3
- 1999 komplettes Neuschreiben des Kerns als PHP4
- 2004 Release PHP5 (aktuell 5.2.x)
- PHP6 im Entwicklungsstadium (seit November 2005)

Wichtige Eigenschaften von PHP

Wichtige Eigenschaften von PHP

- PHP ist eine Interpretersprache.

Wichtige Eigenschaften von PHP

- PHP ist eine Interpretersprache.
- PHP lehnt sich an C/C++, Perl und UNIX® Shells an.

Wichtige Eigenschaften von PHP

- PHP ist eine Interpretersprache.
- PHP lehnt sich an C/C++, Perl und UNIX® Shells an.
- PHP ist meist serverseitige Sprache.

Wichtige Eigenschaften von PHP

- PHP ist eine Interpretersprache.
- PHP lehnt sich an C/C++, Perl und UNIX® Shells an.
- PHP ist meist serverseitige Sprache.
  - Webserver führt PHP Skripte aus.
  - Ausgabe des Skripts geht an Webbrowser.

Wichtige Eigenschaften von PHP

- PHP ist eine Interpretersprache.
- PHP lehnt sich an C/C++, Perl und UNIX® Shells an.
- PHP ist meist serverseitige Sprache.
  - Webserver führt PHP Skripte aus.
  - Ausgabe des Skripts geht an Webbrowser.
- PHP kann aber auch ohne Webserver ausgeführt werden.
  - PHP Interpreter wird lokal aufgerufen.
  - Webserver ist nicht notwendig.

Wichtige Eigenschaften von PHP

- PHP ist eine Interpretersprache.
- PHP lehnt sich an C/C++, Perl und UNIX® Shells an.
- PHP ist meist serverseitige Sprache.
  - Webserver führt PHP Skripte aus.
  - Ausgabe des Skripts geht an Webbrowser.
- PHP kann aber auch ohne Webserver ausgeführt werden.
  - PHP Interpreter wird lokal aufgerufen.
  - Webserver ist nicht notwendig.
- PHP hat eine Vielzahl von Funktionen.
  - 1500+ Funktionen in PHP 5.x.
  - 3000+ Funktionen in PHP 6.
  - Modular erweiterbar.

Koexistenz HTML / PHP

Koexistenz HTML / PHP

Ein einfaches PHP Skript:

```
<html>
<head>
<title>PHP Skript</title>
</head>
<body>


 <script language="php">
  echo "<p>Hallo, Welt!</p>";
  echo "Beim naechsten Ton ist es ".date("h:i:s A")."<br>";
 </script>

</body>
</html>
```

Weitere Methoden PHP zu markieren

Weitere Methoden PHP zu markieren

```
<?php
  echo "<p>Hallo, Welt!</p>";
?>
```

```
<?
  echo "<p>Hallo, Welt!</p>";
?>
```

Wichtig: Jede Anweisung muss mit einem „;“ von einer anderen getrennt werden.

Kommentare im Skript

Kommentare im Skript

Kommentare im Skript werden wie folgt markiert:

```
// Kommentartext (aus C++)

/* Kommentartext (aus C) */

# Kommentartext (aus UNIX Shells und Perl)
```

Es empfiehlt sich aus Gründen der Übersichtlichkeit den Stil nicht zu mischen.

Definition von Konstanten

- PHP erlaubt die Definition von Konstanten.

```
<?php
    define("FOO_BAR", "Do you mean fubar?");
    define('TEST_CONSTANT','Works!');

    echo "Die Konstante heisst: ",FOO_BAR;
?>
```

Konstanten sollten aus Gründen der Übersichtlichkeit immer in Großbuchstaben geschrieben sein.

Variablen in PHP

Variablen in PHP

Regeln und Grundsätze:

- Variablen beginnen mit einem $ Zeichen.
- Sie haben keine Leerzeichen im Namen.
- Buchstaben und Ziffern sind erlaubt.
- Groß-/Kleinschreibung wird unterschieden.
- Keine Sonderzeichen, keine Umlaute:
  - theoretisch möglich
  - praktisch nicht zu empfehlen (übersichtlicher ohne Sonderzeichen)
- Namen dürfen keine PHP-Schlüsselwörter sein.
  - Einzige Ausnahme sind Funktionsnamen.
- Der Unterstrich („underscore”) „_” ist erlaubt.

Variablentypen

Variablentypen

- PHP kennt verschiedene Variablentypen:

Variablentypen

- PHP kennt verschiedene Variablentypen:
  - numerische Variablen (`int`, `double`)
  - Strings (`string`)
  - boolesche Variable (`bool`)
  - Felder (`array`)
  - Klassen/Objekte
  - Resourcen

Variablentypen

- PHP kennt verschiedene Variablentypen:
  - numerische Variablen (`int`, `double`)
  - Strings (`string`)
  - boolesche Variable (`bool`)
  - Felder (`array`)
  - Klassen/Objekte
  - Resourcen
- PHP wählt den Typ *automatisch* abhängig vom Inhalt:

Variablentypen

- PHP kennt verschiedene Variablentypen:
  - numerische Variablen (`int`, `double`)
  - Strings (`string`)
  - boolesche Variable (`bool`)
  - Felder (`array`)
  - Klassen/Objekte
  - Resourcen
- PHP wählt den Typ *automatisch* abhängig vom Inhalt:
  - Fluch *und* Segen.
  - Aufpassen bei Verknüpfungen und Operationen!
  - Vorsicht mit Parametern bei Aufruf von Funktionen!

Variablentypen

- PHP kennt verschiedene Variablentypen:
  - numerische Variablen (`int`, `double`)
  - Strings (`string`)
  - boolesche Variable (`bool`)
  - Felder (`array`)
  - Klassen/Objekte
  - Resourcen
- PHP wählt den Typ *automatisch* abhängig vom Inhalt:
  - Fluch *und* Segen.
  - Aufpassen bei Verknüpfungen und Operationen!
  - Vorsicht mit Parametern bei Aufruf von Funktionen!
- Wichtig: Im Zweifelsfall Datentyp immer explizit angeben!

Implizite Typwandlung

Implizite Typwandlung

- Typen von Variablen lassen sich ändern:
  z.B. `$zahl = (int)$wert;`

Implizite Typwandlung

- Typen von Variablen lassen sich ändern:
  z.B. `$zahl = (int)$wert;`
- PHP wandelt dabei den Inhalt der Variable um.

Implizite Typwandlung

- Typen von Variablen lassen sich ändern:
  z.B. `$zahl = (int)$wert;`
- PHP wandelt dabei den Inhalt der Variable um.
- PHP macht dieses sogenannte *Type Casting* auch automatisch:

```
<?php
  $foo = "0";                          // $foo is string (ASCII 48)
  $foo += 2;                           // $foo is now an integer (2)
  $foo = $foo + 1.3;                   // $foo is now a float (3.3)
  $foo = 5 + "10 Little Piggies"; // $foo is integer (15)
  $foo = 5 + "10 Small Pigs";     // $foo is integer (15)
?>
```

Aufpassen!

So entstehen Sicherheitslücken und schwer zu findende Fehler!

Umgang mit Variablen

- Variablen immer mit Defaultwerten initialisieren.
- Inhalt immer auf Plausibilität prüfen.
- Externe Daten immer sorgfältig prüfen.
  - Extern ist alles, was das Skript einliest oder bekommt.
  - Extern ist
    - jede Datei,
    - jede Webseite,
    - jegliche Daten vom Browser und damit
    - jegliche Benutzereingaben.
- Ungültige Daten in Variablen
  - immer mit Fehler abfangen oder
  - durch Ersatzwerte korrigieren.

Typen testen

Typen testen

- Variablen lassen sich auf Datentyp prüfen:

Typen testen

- Variablen lassen sich auf Datentyp prüfen:
  - `is_array()`
  - `is_bool()`
  - `is_double()`
  - `is_float()`
  - `is_integer()`
  - `is_numeric()`
  - `is_string()`

Typen testen

- Variablen lassen sich auf Datentyp prüfen:
  - `is_array()`
  - `is_bool()`
  - `is_double()`
  - `is_float()`
  - `is_integer()`
  - `is_numeric()`
  - `is_string()`
- Komplette Aufzählung steht in PHP Dokumentation.

Strings in PHP

Strings in PHP

- Strings werden mit
  - einfachen Anführungszeichen oder
  - doppelten Anführungszeichen

  markiert.

Strings in PHP

- Strings werden mit
  - einfachen Anführungszeichen oder
  - doppelten Anführungszeichen

  markiert.
- PHP übernimmt Inhalte zwischen einfachen Anführungszeichen unverändert.

Strings in PHP

- Strings werden mit
  - einfachen Anführungszeichen oder
  - doppelten Anführungszeichen

  markiert.
- PHP übernimmt Inhalte zwischen einfachen Anführungszeichen unverändert.
- PHP interpretiert Inhalte zwischen doppelten Anführungszeichen.

Strings in PHP

- Strings werden mit
  - einfachen Anführungszeichen oder
  - doppelten Anführungszeichen

  markiert.
- PHP übernimmt Inhalte zwischen einfachen Anführungszeichen unverändert.
- PHP interpretiert Inhalte zwischen doppelten Anführungszeichen.

```
<?
    echo '<a href="' . $uri . '">' . $text . '</a>';
    echo "<a href=\"$uri\">$text</a>";
?>
```

Felder / Arrays

Felder / Arrays

Felder dienen zum Abbilden von Reihen:

```
<?php
   // Erstellen eines Arrays
   $diskspace[0] = 39.4;
   $diskspace[1] = 34.5;
   $diskspace[2] = 32.1;
   // Oder auch so
   $diskspace = array(39.4, 34.5, 32.1);
?>
```

Indizes können beliebig sein.

Mehrdimensionale Arrays

Mehrdimensionale Arrays

Darf's eine Dimension mehr sein?

```
<?php
   // Wie in einer Dimension,
   // nur eben anders
   $farbacker = array( array( "rot", "gruen", "blau"),
                       array( "cyan", "gelb", "braun"),
                       array( "lila", "violett", "purpur"));
?>
```

Dimensionen sind syntaktisch nicht limitiert.

Größe von Arrays

Größe von Arrays

- Größe von Arrays läßt sich durch `count()` bestimmen.

Größe von Arrays

- Größe von Arrays läßt sich durch `count()` bestimmen.
- Ergebnis gibt
  - Anzahl der Elemente im Array,
  - 0 für leeren Array oder nicht gesetzte Variable bzw.
  - 1 wenn das Argument kein Array oder nicht zählbar ist zurück.

Größe von Arrays

- Größe von Arrays läßt sich durch `count()` bestimmen.
- Ergebnis gibt
  - Anzahl der Elemente im Array,
  - 0 für leeren Array oder nicht gesetzte Variable bzw.
  - 1 wenn das Argument kein Array oder nicht zählbar ist zurück.
- Man kann ebenso `sizeof()` verwenden.

Assoziative Arrays

Assoziative Arrays

Zugriff auf Feldelemente erfolgt mittels eines Schlüssels:

Assoziative Arrays

Zugriff auf Feldelemente erfolgt mittels eines Schlüssels:

```
<?php
$woche = array( "Montag" => "schlecht",
                "Mittwoch" => "besser",
                "Freitag" => "frei" );

// Ausgabe: besser
echo $woche['Mittwoch'];
?>
```

Variablentests

Variablentests

- PHP kennt folgende Funktionen:
  - `isset()` - prüft, ob eine Variable existiert.
  - `unset()` - löscht eine Variable.
  - `empty()` - prüft, ob eine Variable leer ist.

Variablentests

- PHP kennt folgende Funktionen:
  - `isset()` - prüft, ob eine Variable existiert.
  - `unset()` - löscht eine Variable.
  - `empty()` - prüft, ob eine Variable leer ist.
- Wichtig für
  - das Abfangen von Benutzereingaben und
  - das Setzen von Variablen auf definierte Werte.

Operatoren

Operatoren

PHP kennt eine Reihe von Operatoren:

Operatoren

PHP kennt eine Reihe von Operatoren:

- Inkrement-/Dekrementoperator: `++ --`
- arithmetische Operatoren: `+ - * / %`
- Zuweisungsoperatoren: `= += -= *= /= %=`
- Stringoperatoren: `. .=`
- vergleichende Operatoren: `== === != !== < <= > >=`
- logische Operatoren: `&& and || or ! xor`
- bitweise Operatoren: `& | << >> ^`

if/else Abfragen

if/else Abfragen

Was wäre wenn. . .

if/else Abfragen

Was wäre wenn...

```
<?php
    if ( $anzahl_kaffee >= 15 ) {
        echo "Danke, nein.<br>";
    }
    else {
        echo "Ja, noch einen.<br>";
    }
?>
```

if/elseif Abfragen

Was wäre wenn. . .und wenn nicht. . .

```
<?php
    if ( $anzahl_kaffee >= 15 ) {
        echo "Danke, nein.<br>";
    }
    elseif ( $zucker > 0 ) {
        echo "Ja, noch einen.<br>";
    }
    else {
         echo "Nein, Danke!<br">;
    }
?>
```

for() Schleife

Die `for()` Schleife stammt aus der Programmiersprache C:

for() Schleife

Die `for()` Schleife stammt aus der Programmiersprache C:

```
<?php
  echo "<table>";
  for ( $zaehler=0; $zaehler<200; $zaehler++ ) {
     echo "<tr><td>Zeile</td><td>$zaehler</td></tr>";
  }
  echo "</table>";
?>
```

```
for ( Initialisierung;
      Schleifenbedingung;
      Aenderungssequenz ) {
    ...
}
```

foreach() - Durchlaufen von Arrays

foreach() - Durchlaufen von Arrays

`foreach` erlaubt das Durchlaufen eines Arrays:

```
<?php
$feld = array( 1, 2, 3, 17 );

/* Nur fuer die Darstellung der Indizes */
$i = 0;

foreach ($feld as $element) {
    echo "\$feld[$i] => $element.\n";
    $i++;
}
?>
```

while() Schleife

Die `while()` Schleife hat die Schleifenbedingung am Anfang:

while() Schleife

Die `while()` Schleife hat die Schleifenbedingung am Anfang:

```
<select name="whatever">
<?php
while ($data = get_my_data($requeteID))
{
  $menu .= "<option value=\"$data['id']\"";
  $menu .= ($data['id'] == $_GET['id'] ? ' selected>' :'>');
  $menu .= $data['name'].'</option>';
}
echo $menu;
?>
</select>
```

Wenn die Bedingung nicht erfüllt ist, dann wird die Schleife nicht durchlaufen.

do/while() Schleife

Die `do/while()` Schleife hat mindestens einen Durchlauf.

do/while() Schleife

Die `do/while()` Schleife hat mindestens einen Durchlauf.

```
<?php
$i = 0;
do {
        echo $i;
} while ($i > 0);
?>
```

Schleife wird nur einmal durchlaufen, weil sich `$i` nicht aendert.

Einbinden und Auslagern

Einbinden und Auslagern

- PHP kann Teile anderer Dateien einfügen:

Einbinden und Auslagern

- PHP kann Teile anderer Dateien einfügen:
  - `include()` - Einbinden während der Laufzeit
  - `include_once()` - Einbinden während der Laufzeit
  - `require()` - Einbinden bevor Skript läuft

Einbinden und Auslagern

- PHP kann Teile anderer Dateien einfügen:
  - `include()` - Einbinden während der Laufzeit
  - `include_once()` - Einbinden während der Laufzeit
  - `require()` - Einbinden bevor Skript läuft
- Damit können HTML-Blöcke und Skripte ausgelagert werden.

Einbinden und Auslagern

- PHP kann Teile anderer Dateien einfügen:
  - `include()` - Einbinden während der Laufzeit
  - `include_once()` - Einbinden während der Laufzeit
  - `require()` - Einbinden bevor Skript läuft
- Damit können HTML-Blöcke und Skripte ausgelagert werden.
- Funktionen führen eingebundenen PHP Code aus.

Einbinden und Auslagern

- PHP kann Teile anderer Dateien einfügen:
  - `include()` - Einbinden während der Laufzeit
  - `include_once()` - Einbinden während der Laufzeit
  - `require()` - Einbinden bevor Skript läuft
- Damit können HTML-Blöcke und Skripte ausgelagert werden.
- Funktionen führen eingebundenen PHP Code aus.
  - Niemals Code aus dem Netz einbetten!
  - Niemals URLs als Argument verwenden!

  So entstehen Sicherheitslücken!

require() Beispiel

require() Beispiel

Laden von Funktionen

```
<?php
    require("funktionen.php");

    // Aufruf einer extern definierten Funktion
    $bildname = hole_dateinamen($url);
    echo "Das Bild heisst $bildname.";
?>
```

include() Beispiel

include() Beispiel

Auslagern von Skriptteilen

```
<?php
   // ...*lotsoffancystuffhere*...

   if ( $style == 1 ) {
      include("style_blue.php");
   }
   else {
      include("style_default.php");
   }
?>
```

Klassen in PHP

Klassen in PHP

- Klassen sind ein „Container” für
  - Funktionen (Methoden) und
  - Attribute.

  Klassen sind letztlich selbstdefinierte Typen.

Klassen in PHP

- Klassen sind ein „Container” für
  - Funktionen (Methoden) und
  - Attribute.

  Klassen sind letztlich selbstdefinierte Typen.
- Klassen sind „Baupläne” für Objekte (oder Objektinstanzen).

Klassen in PHP

- Klassen sind ein „Container” für
  - Funktionen (Methoden) und
  - Attribute.

  Klassen sind letztlich selbstdefinierte Typen.
- Klassen sind „Baupläne” für Objekte (oder Objektinstanzen).
- Methoden derselben Klasse sind verwandt.
- Methoden derselben Klasse können Daten untereinander teilen.

Klassen in PHP

- Klassen sind ein „Container” für
  - Funktionen (Methoden) und
  - Attribute.

  Klassen sind letztlich selbstdefinierte Typen.
- Klassen sind „Baupläne” für Objekte (oder Objektinstanzen).
- Methoden derselben Klasse sind verwandt.
- Methoden derselben Klasse können Daten untereinander teilen.
- Klassen werden oft in seperate Dateien gespeichert:

Klassen in PHP

- Klassen sind ein „Container” für
  - Funktionen (Methoden) und
  - Attribute.

  Klassen sind letztlich selbstdefinierte Typen.
- Klassen sind „Baupläne” für Objekte (oder Objektinstanzen).
- Methoden derselben Klasse sind verwandt.
- Methoden derselben Klasse können Daten untereinander teilen.
- Klassen werden oft in seperate Dateien gespeichert:
  - Einbinden mittels `include()`
  - Code kann leichter wiederverwendet werden
  - Skripte werden übersichtlicher

Beispiel: Definition einer Klasse

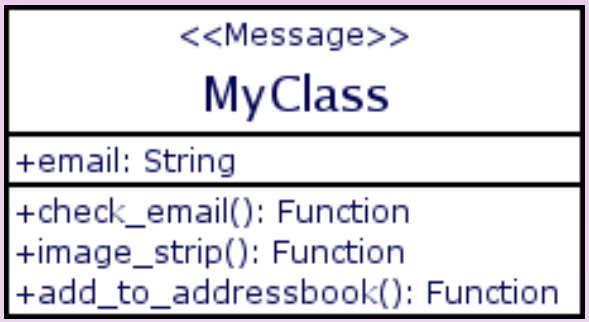
Beispiel: Definition einer Klasse

```
<?php
class MyClass{
    var $email;

    // Use a function without variables
    function check_email(){
        if(ereg("^.+@.+\..+$", $this->email))
            return (true);
        else
            return (false);
    }

    // Use a function with variables
    function image_strip($somehtml){
        $somehtml = preg_replace("/(<img)(.*?)(>)/si", "", $somehtml);
        return $somehtml;
    }
}
?>
```

Die definierte Klasse als UML Diagramm



Die Klasse ist nun ein eigener Datentyp, welcher einen String und drei Funktionen enthält, die auf ihn wirken bzw. mit ihm Operationen durchführen. Mit Klassen lassen sich komplizierte Datenstrukturen besser abbilden.

Beispiel: Verwenden von Klassen

Beispiel: Verwenden von Klassen

- Einbinden der Definition mittels `include()`.

Beispiel: Verwenden von Klassen

- Einbinden der Definition mittels `include()`.
- Anlegen eines Klassenobjekts:

```
$myclass = new MyClass();
```

Beispiel: Verwenden von Klassen

- Einbinden der Definition mittels `include()`.
- Anlegen eines Klassenobjekts:
  ```
  $myclass = new MyClass();
  ```
- Belegen der internen Variable:
  ```
  $myclass->email = 'checkme@example.net';
  ```

Beispiel: Verwenden von Klassen

- Einbinden der Definition mittels `include()`.
- Anlegen eines Klassenobjekts:
  ```
  $myclass = new MyClass();
  ```
- Belegen der internen Variable:
  ```
  $myclass->email = 'checkme@example.net';
  ```
- Prüfen der E-Mail-Adresse:
  ```
  $check_email = $myclass->check_email();
  ```
- `$check_email` enthält nun *wahr* oder *falsch*

Mehr über Klassen

Interaktion mit Clients

Interaktion mit Clients

- Server und Webbrowser kommunizieren.

Interaktion mit Clients

- Server und Webbrowser kommunizieren.
- Aufrufen von Seiten bedeutet:
  - Webseite $\longrightarrow$ Benutzer

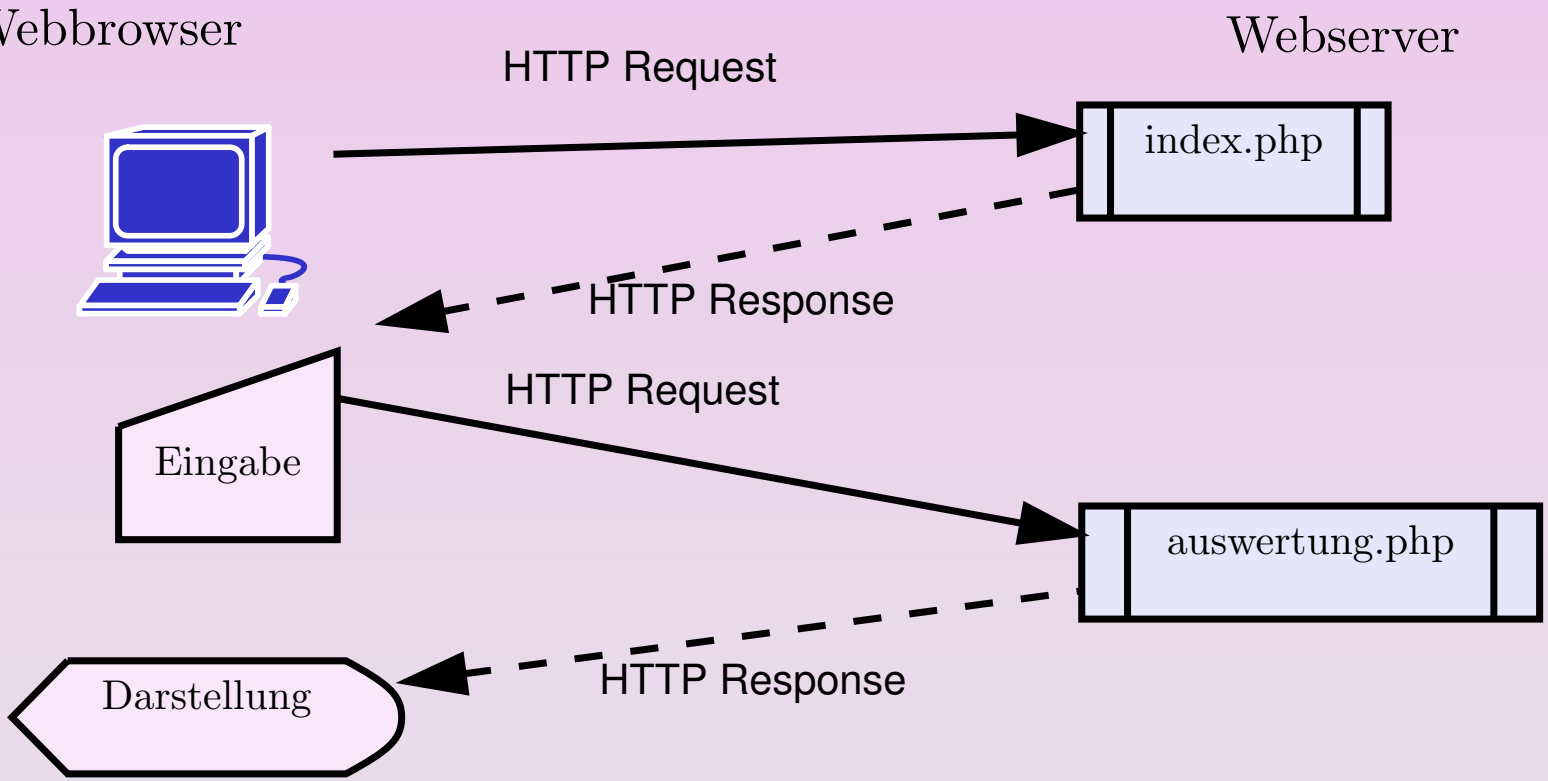
Interaktion mit Clients

- Server und Webbrowser kommunizieren.
- Aufrufen von Seiten bedeutet:
  - Webseite $\longrightarrow$ Benutzer
- Eingabe von Daten:
  - beispielsweise durch Webformular
  - Benutzer $\longrightarrow$ Webseite

Interaktion mit Clients

- Server und Webbrowser kommunizieren.
- Aufrufen von Seiten bedeutet:
  - Webseite $\longrightarrow$ Benutzer
- Eingabe von Daten:
  - beispielsweise durch Webformular
  - Benutzer $\longrightarrow$ Webseite
- PHP kann Daten empfangen und verarbeiten.

Interaktion mit Clients - Skriptaufrufe



Daten mittels GET übertragen

Daten mittels GET übertragen

- `http://du.da.at/cd.php?artikel=42&anzahl=7`
  - ruft das Skript cd.php auf,
  - übermittelt alles nach dem „?” an Server
  - und füllt vordefinierte Variablen.

Daten mittels GET übertragen

- `http://du.da.at/cd.php?artikel=42&anzahl=7`
  - ruft das Skript cd.php auf,
  - übermittelt alles nach dem „?” an Server
  - und füllt vordefinierte Variablen.
- Daten werden in *Superglobals* gespeichert.
  - Superglobals können auch Felder sein.
  - `$_GET['artikel']` enthält ’42’.
  - `$_GET['anzahl']` enthält ’7’.

Daten mittels GET übertragen

- `http://du.da.at/cd.php?artikel=42&anzahl=7`
  - ruft das Skript cd.php auf,
  - übermittelt alles nach dem „?” an Server
  - und füllt vordefinierte Variablen.
- Daten werden in *Superglobals* gespeichert.
  - Superglobals können auch Felder sein.
  - `$_GET['artikel']` enthält ’42’.
  - `$_GET['anzahl']` enthält ’7’.
- `$_GET[]` wird pro Skriptaufruf neu belegt.

Daten mittels POST übertragen

Daten mittels POST übertragen

Die POST Methode wird oft in Webformularen verwendet:

```
<html>
<head><title>Blarg</title></head>
<body>
<form name="" action="giveme.php" method="POST">
   Album <input type="text" name="album"><br>
   Gruppe <input type="text" name="gruppe"><br>
   <input type="submit" value="Senden">
</form>
</body>
</html>
```

Daten mittels POST übertragen

Die POST Methode wird oft in Webformularen verwendet:

```
<html>
<head><title>Blarg</title></head>
<body>
<form name="" action="giveme.php" method="POST">
   Album <input type="text" name="album"><br>
   Gruppe <input type="text" name="gruppe"><br>
   <input type="submit" value="Senden">
</form>
</body>
</html>
```

klick

Daten mittels POST übertragen (2)

Daten mittels POST übertragen (2)

- Die Daten werden ebenso in ein Superglobal Array gespeichert.

Daten mittels POST übertragen (2)

- Die Daten werden ebenso in ein Superglobal Array gespeichert.
- `$_POST['album']`
- `$_POST['gruppe']`

Daten mittels POST übertragen (2)

- Die Daten werden ebenso in ein Superglobal Array gespeichert.
- `$_POST['album']`
- `$_POST['gruppe']`
- Das Attribut „name” im <input> Tag definiert den Index.

Daten mittels POST übertragen (2)

- Die Daten werden ebenso in ein Superglobal Array gespeichert.
- `$_POST['album']`
- `$_POST['gruppe']`
- Das Attribut „name” im <input> Tag definiert den Index.
- `$_POST[]` wird pro Aufruf ebenso neu belegt.

PHP Superglobals

PHP Superglobals

- Es gibt noch andere Superglobals, die externe Daten enthalten.

PHP Superglobals

- Es gibt noch andere Superglobals, die externe Daten enthalten.
- `$_GET[]` und `$_POST[]`
- `$_COOKIE[]` - Cookiedaten.
- `$_FILES[]` - Daten von Dateiuploads.
- `$_SERVER[]` - Infos weitergereicht vom Webserver.
- `$_ENV[]` - Umgebung des Webservers.
- `$_SESSION[]` - registrierte Sessiondaten.
- `$_REQUEST[]` - Kombination aus `$_GET[]`, `$_POST[]` und `$_COOKIE[]`

PHP Superglobals

- Es gibt noch andere Superglobals, die externe Daten enthalten.
- `$_GET[]` und `$_POST[]`
- `$_COOKIE[]` - Cookiedaten.
- `$_FILES[]` - Daten von Dateiuploads.
- `$_SERVER[]` - Infos weitergereicht vom Webserver.
- `$_ENV[]` - Umgebung des Webservers.
- `$_SESSION[]` - registrierte Sessiondaten.
- `$_REQUEST[]` - Kombination aus `$_GET[]`, `$_POST[]` und `$_COOKIE[]`
- Mittels `phpinfo()` kann man Superglobals testweise anzeigen lassen.

Umgang mit Superglobals und Daten

Umgang mit Superglobals und Daten

- Alle Daten sind schuldig bis zum Beweis des Gegenteils!

Umgang mit Superglobals und Daten

- Alle Daten sind schuldig bis zum Beweis des Gegenteils!
- Lieber Daten ablehnen als Daten filtern!

Umgang mit Superglobals und Daten

- Alle Daten sind schuldig bis zum Beweis des Gegenteils!
- Lieber Daten ablehnen als Daten filtern!
- Daten werden bei Übergabe und vor jedem Verarbeitungsschritt geprüft!

Umgang mit Superglobals und Daten

- Alle Daten sind schuldig bis zum Beweis des Gegenteils!
- Lieber Daten ablehnen als Daten filtern!
- Daten werden bei Übergabe und vor jedem Verarbeitungsschritt geprüft!
- Niemals direkt Kommandos vom Client/User annehmen!

Umgang mit Superglobals und Daten

- Alle Daten sind schuldig bis zum Beweis des Gegenteils!
- Lieber Daten ablehnen als Daten filtern!
- Daten werden bei Übergabe und vor jedem Verarbeitungsschritt geprüft!
- Niemals direkt Kommandos vom Client/User annehmen!
- Vorsicht vor Sonderzeichen!

Umgang mit Superglobals und Daten

- Alle Daten sind schuldig bis zum Beweis des Gegenteils!
- Lieber Daten ablehnen als Daten filtern!
- Daten werden bei Übergabe und vor jedem Verarbeitungsschritt geprüft!
- Niemals direkt Kommandos vom Client/User annehmen!
- Vorsicht vor Sonderzeichen!
- Code muß im Zweifelsfalle Aktionen verbieten!

Umgang mit Superglobals und Daten

- Alle Daten sind schuldig bis zum Beweis des Gegenteils!
- Lieber Daten ablehnen als Daten filtern!
- Daten werden bei Übergabe und vor jedem Verarbeitungsschritt geprüft!
- Niemals direkt Kommandos vom Client/User annehmen!
- Vorsicht vor Sonderzeichen!
- Code muß im Zweifelsfalle Aktionen verbieten!
- Filter nur einsetzen, wenn man sie versteht!

Umgang mit Superglobals und Daten

- Alle Daten sind schuldig bis zum Beweis des Gegenteils!
- Lieber Daten ablehnen als Daten filtern!
- Daten werden bei Übergabe und vor jedem Verarbeitungsschritt geprüft!
- Niemals direkt Kommandos vom Client/User annehmen!
- Vorsicht vor Sonderzeichen!
- Code muß im Zweifelsfalle Aktionen verbieten!
- Filter nur einsetzen, wenn man sie versteht!
- Verstehen der Daten hilft bei deren Verarbeitung!

Nützlicher Tip für Webformulare

- Formulare müssen immer in *action=* ein Skript angeben.
- URL dieses Skriptes muß
  - absolut oder
  - relativ angegeben werden.

Nützlicher Tip für Webformulare

- Formulare müssen immer in *action=* ein Skript angeben.
- URL dieses Skriptes muß
  - absolut oder
  - relativ angegeben werden.
- `$_SERVER['PHP_SELF']` ist immer das aufgerufene Skript.
- `action="<?php echo $_SERVER['PHP_SELF']; ?>"` in Formularen verwenden.
- Jedes Skript wertet seine eigenen Formulare aus.

Beispiel: GET im Farbeinsatz

Beispiel: GET im Farbeinsatz

Gesucht:
Ein PHP Skript welches durch Aufruf die Farben der Webseite setzt:

`http://du.da.at/farbe.php?bgcolor=000000&text=aaef70`

Bitte jetzt anfangen Code zu schreiben:
Man nehme einen Editor, entwerfe ein PHP-Skript und teste es auf dem lokalen Webserver.

Beispiel: Messen von Zeit

Beispiel: Messen von Zeit

Wie schnell ist ein Skript?

```
<?php
// time() liefert die Sekunden seit
// Beginn der UNIX Epoche, sprich
// "January 1 1970 00:00:00 GMT"
$start = time();
// Here be commands
$end = time();
echo "Das Skript lief ".($end-$start).
     " Sekunden.<br>";
?>
```

Siehe dazu auch die Funktion `microtime()`.

Beispiel: Formulareingabe

Beispiel: Formulareingabe

- Aufbau eines Formulares mit den folgenden Eingabefeldern:
  - Vorname
  - Name
  - Straße
  - Postleitzahl
  - Ort
  - Telefonnummer
  - E-Mail-Adresse
- Skript soll Daten in Empfang nehmen und ausgeben.
- Skript muß Inhalte der Eingabefelder überprüfen!
  - Testen, ob Eingaben dem richtigen Typ entsprechen.
  - Längenbegrenzungen.

Hinweis: Generieren von URLs

Hinweis: Generieren von URLs

- URLs haben ein definiertes Format (RFC 1738).

Hinweis: Generieren von URLs

- URLs haben ein definiertes Format (RFC 1738).
- Nicht jeder String ist „URL-tauglich”.

Hinweis: Generieren von URLs

- URLs haben ein definiertes Format (RFC 1738).
- Nicht jeder String ist „URL-tauglich”.
- PHP kennt Stringkonversionsfunktionen:
  - `urlencode()` - kodiert alle nichtalphanumerischen Zeichen außer `-_.`
  - `rawurlencode()` - kodiert gemäß RFC 1738
  - `htmlentities()` - kodiert alle Sondezeichen in ihre HTML-Notation.
  - `htmlspecialchars()` - kodiert bestimmte Sondezeichen in ihre HTML-Notation.
- Mit diesen Funktionen können generierte URLs „entschärft” werden, wenn sie Daten enthalten, die nicht unter der Kontrolle des Skripts stehen.

Beispiel: Umleiten durch HTTP Header

Beispiel: Umleiten durch HTTP Header

Umleiten des Browsers via HTML:

```
<meta http-equiv="refresh" content="0; URL=URI">
```

Beispiel: Umleiten durch HTTP Header

Umleiten des Browsers via HTML:

```
<meta http-equiv="refresh" content="0; URL=URI">
```

Umleiten kann auch via HTTP Header erfolgen:

```
<?php
   if ( $redirect ) {
      header("Location: http://www.disney.com/");
      exit;
   }
?>
<html>
<!-- Here be content -->
</html>
```

Wichtig: `header()` funktioniert nur, wenn vorher **keine** Ausgabe erfolgte!

Dateioperationen - Abspeichern von Daten

Dateioperationen - Abspeichern von Daten

```
<?php
   $fc = Array( "1. Zeile", "2. Zeile",
                "3. Zeile" );
   $handle = fopen("daten.dat","w");
   // Schreibe Feld $fc in Datei
   foreach($fc as $line)
   {
      fputs($handle,$line);
   }
   fclose($handle);
?>
```

Stringoperationen explode() und implode()

Stringoperationen explode() und implode()

- PHP kennt eine Vielzahl von Stringfunktionen

Stringoperationen explode() und implode()

- PHP kennt eine Vielzahl von Stringfunktionen
- Wandeln von Feldern in Strings und umgekehrt
  - `implode()` - baut aus Array String zusammen
  - `explode()` - nimmt String auseinander

Stringoperationen explode() und implode()

- PHP kennt eine Vielzahl von Stringfunktionen
- Wandeln von Feldern in Strings und umgekehrt
  - `implode()` - baut aus Array String zusammen
  - `explode()` - nimmt String auseinander
- Nützlich für CSV (Comma Seperated Values) Dateien

Beispiel: implode()

Beispiel: implode()

```
<?php
// Array mit Inhalt erzeugen
$feld = array('Nachname', 'E-Mail', 'Fon');
// String zusammenbauen
$comma_separated = implode(",", $feld);

// Ausgabe: Nachname,E-Mail,Fon
echo $comma_separated;

?>
```

Beispiel: explode()

Beispiel: explode()

```
<?php
    $pizza = "Schinken Kaese Ei Spinat";
    $belag = explode(" ", $pizza);
    echo $belag[0]; // Schinken
    echo $belag[1]; // Kaese
    echo $belag[2]; // Ei
    echo $belag[3]; // Spinat
?>
```

Vergleichen von Strings

Vergleichen von Strings

- Stringvergleiche werden durch eigene Funktionen ausgeführt:
  - `strcmp()` - vergleicht Strings mit Groß-/Kleinschreibung.
  - `strcasecmp()` - vergleicht Strings ohne Groß-/Kleinschreibung.
- Rückgabewert gibt Ergebnis an:
  - `strcmp( $a, $b )`
  - `0` - Strings sind gleich
  - `<0` - `$a` ist größer als `$b`
  - `>0` - `$a` ist kleiner als `$b`

Kommunikation mit Datenbanken

Datenbankunterstützung in PHP

Datenbankunterstützung in PHP

- PHP unterstützt eine Reihe von Datenbanken
  - Berkeley DB
  - Microsoft® SQL
  - MySQL®
  - Oracle®
  - Postgres
  - SQLite Embeddable SQL Database
  - ...

Datenbankunterstützung in PHP

- PHP unterstützt eine Reihe von Datenbanken
  - Berkeley DB
  - Microsoft® SQL
  - MySQL®
  - Oracle®
  - Postgres
  - SQLite Embeddable SQL Database
  - ...
- Verfügbarkeit hängt von PHP Installation ab!
  - Vor Erstellen des Codes planen!
  - Abstimmen mit Systemadministration!

SQL - CREATE TABLE

SQL - CREATE TABLE

```
CREATE DATABASE music;

USE music;

CREATE TABLE musiclink (
    ID       INT,
    Created DATETIME,
    Band     VARCHAR(128),
    Album    VARCHAR(128),
    URL      VARCHAR(160),
    PRIMARY KEY ('ID','URL')
);
```

SQL - INSERT

SQL - INSERT

Einfügen von Daten

```
INSERT INTO musiclink
 (Created,Band,Album,URL)
 VALUES
 (NOW(),'The Ghost of Lemora',
  'Reach for the Ground',
  'http://www.theghostoflemora.co.uk/');
```

SQL - SELECT

SQL - SELECT

Auslesen von Daten

```
SELECT * FROM musiclink WHERE
  Album='Floodland';

SELECT URL,Album FROM musiclink WHERE
  Band='Scary Bitches';
```

SQL - SELECT verfeinert

SQL - SELECT verfeinert

Auslesen und Suchen von Daten

```
SELECT URL FROM musiclink WHERE
  Band LIKE '%laibach%';

SELECT Band,URL FROM musiclink WHERE
  Album LIKE 'The%';
```

SQL - UPDATE

SQL - UPDATE

Modifizieren von Daten

```
UPDATE musiclink SET Band='In Extremo'
  WHERE ID=23;

UPDATE musiclink
  SET
     Band='- Sold out -',
     URL ='http://shop.at/nixmehrda.html'
  WHERE Album LIKE 'The%' LIMIT 1;
```

SQL - UPDATE

Modifizieren von Daten

```
UPDATE musiclink SET Band='In Extremo'
  WHERE ID=23;

UPDATE musiclink
  SET
     Band='- Sold out -',
     URL ='http://shop.at/nixmehrda.html'
  WHERE Album LIKE 'The%' LIMIT 1;
```

Wichtig: Aufpassen auf Wildcards! In Zweifelsfällen `LIMIT 1` verwenden.

Verbinden zu Datenbanken in PHP

Verbinden zu Datenbanken in PHP

- Zwei Schnittstellen zur MySQL DB:
  - MySQL Funktionen
  - MySQL Improved Funktionen (MySQLi)

Verbinden zu Datenbanken in PHP

- Zwei Schnittstellen zur MySQL DB:
  - MySQL Funktionen
  - MySQL Improved Funktionen (MySQLi)
  - Wichtig: Beide haben leicht unterschiedliche Parameterreihenfolge!

Verbinden zu Datenbanken in PHP

- Zwei Schnittstellen zur MySQL DB:
  - MySQL Funktionen
  - MySQL Improved Funktionen (MySQLi)
  - Wichtig: Beide haben leicht unterschiedliche Parameterreihenfolge!
- MySQLi Schnittstelle ab MySQL 4.1.3 verfügbar

Verbinden zu Datenbanken in PHP

- Zwei Schnittstellen zur MySQL DB:
  - MySQL Funktionen
  - MySQL Improved Funktionen (MySQLi)
  - Wichtig: Beide haben leicht unterschiedliche Parameterreihenfolge!
- MySQLi Schnittstelle ab MySQL 4.1.3 verfügbar
- MySQLi unterstützt zusätzliche Optionen der MySQL DB

Verbinden zu Datenbanken in PHP

- Zwei Schnittstellen zur MySQL DB:
  - MySQL Funktionen
  - MySQL Improved Funktionen (MySQLi)
  - Wichtig: Beide haben leicht unterschiedliche Parameterreihenfolge!
- MySQLi Schnittstelle ab MySQL 4.1.3 verfügbar
- MySQLi unterstützt zusätzliche Optionen der MySQL DB
- MySQLi besonders wichtig bei Kodierung von Zeichen
  - UTF-8
  - ISO-8859-15
  - ISO-8859-2
  - ...

Verbinden zu Datenbanken in PHP (2)

Verbinden zu Datenbanken in PHP (2)

```
<?php
    $db_user     = 'murtaugh';
    $db_pass     = 'jo5pesc1';
    $db_database = 'depot';
    $db_host     = '127.0.0.1';
    $db_port     = '3306';
    $dblink = mysqli_connect($db_host,$db_user,
              $db_pass,$db_database,$db_port);
    if ( $dblink ) {
      // Verbindung erfolgreich
    }
    else {
      print "Fehler: ".mysqli_connect_error();
      exit(10);
    }
?>
```

Auswählen einer Datenbank

Auswählen einer Datenbank

```
<?php
   // Verbinden
   $dblink = mysqli_connect($db_host, $db_user,
                            $db_pass);

   // Datenbank "foobar" auswaehlen
   // Entspricht "USE foobar;"
   $selected = mysqli_select_db($dblink, 'foobar');
   if (!$selected) {
      die ('Cannot use DB : '.mysqli_error($dblink));
   }
   mysqli_set_charset($dblink, "utf8");
?>
```

SQL-Kommandos absetzen

SQL-Kommandos absetzen

Wie kommt das SQL zur Datenbank?

```
<?php
    $sql = "CREATE DATABASE user_xxx";
    $result = mysqli_query($dblink,$sql);
    if ( !$result ) {
        die('Ups, Fehler: '.mysqli_error($dblink));
    }

    // Hier geht es weiter
?>
```

Längere SQL-Kommandos (1)

Längere SQL-Kommandos (1)

```
<?php
   $sql  = "CREATE TABLE kontakt (";
   $sql .= "ID INT auto_increment,";
   $sql .= "Vorname VARCHAR(100),";
   $sql .= "Name VARCHAR(100),";
   $sql .= "Email VARCHAR(120)";
   $sql .= ")";
   $result = mysqli_query($dblink,$sql);
   if ( !$result ) {
      die('Ups, Fehler: '.mysqli_error($dblink));
   }
?>
```

Längere SQL-Kommandos (2)

```
<?php
   $sql  = "INSERT INTO kontakt ";
   $sql .= "(Vorname,Name,Email) ";
   $sql .= "VALUES ";
   $sql .= "('Lynx','Pfeiffer',";
   $sql .= "'lynx@luchs.at')";
   $result = mysqli_query($dblink,$sql);
   if ( !$result ) {
      die('Ups, Fehler: '.mysqli_error($dblink));
   }
?>
```

Längere SQL-Kommandos - sprintf

```
<?php
   // Format definieren
   $frmt  = "INSERT INTO kontakt ";
   $frmt .= "(Vorname,Name,Email) ";
   $frmt .= "VALUES ('%s','%s','%s')";
   // SQL String mit Daten fuellen
   $sql   = sprintf( $frmt, $_POST['vorname'],
            $_POST['name'], $_POST['email'] );
   $result = mysqli_query($dblink,$sql);
   if ( !$result ) {
      die('Ups, Fehler: '.mysqli_error($dblink));
   }
?>
```

Längere SQL-Kommandos - sprintf

```
<?php
   // Format definieren
   $frmt  = "INSERT INTO kontakt ";
   $frmt .= "(Vorname,Name,Email) ";
   $frmt .= "VALUES ('%s','%s','%s')";
   // SQL String mit Daten fuellen
   $sql   = sprintf( $frmt, $_POST['vorname'],
            $_POST['name'], $_POST['email'] );
   $result = mysqli_query($dblink,$sql);
   if ( !$result ) {
      die('Ups, Fehler: '.mysqli_error($dblink));
   }
?>
```

Wichtig: Der Inhalt von `$frmt` muß unter Kontrolle des Skripts bleiben!

Auslesen von Daten

Auslesen von Daten

```
<?php
   // Abfrage zusammenbauen
   $sql  = "SELECT Vorname,Name,Email ";
   $sql .= "FROM kontakt LIMIT 15";
   $result = mysqli_query($dblink,$sql);

   // Wenn Ergebnisse vorhanden, dann diese
   // holen und ausgeben
   if ( $result ) {
      while ($row = mysqli_fetch_array($result,MYSQL_ASSOC)) {
         echo "Vorname: ".$row['Vorname']."<br>";
         echo "Name: ".$row['Name']."<br>";
         echo "E-Mail: ".$row['Email']."<br>";
      }
   }
?>
```

Bereinigen von SQL Statements

Bereinigen von SQL Statements

- PHP Variablen dürfen **nie** ungeprüft in SQL Strings!
- Bestimmte Zeichen verändern das SQL Statement.

Bereinigen von SQL Statements

- PHP Variablen dürfen **nie** ungeprüft in SQL Strings!
- Bestimmte Zeichen verändern das SQL Statement.
- Bereinigen geschieht
  1. automatisch durch PHPs „Magic Quotes” Modus oder
  2. durch `mysqli_real_escape_string()`.

Bereinigen von SQL Statements

- PHP Variablen dürfen **nie** ungeprüft in SQL Strings!
- Bestimmte Zeichen verändern das SQL Statement.
- Bereinigen geschieht
  1. automatisch durch PHPs „Magic Quotes” Modus oder
  2. durch `mysqli_real_escape_string()`.
- Beide Methoden sind unabhängig voneinander!
  - Kann zu doppeltem „Escapen” führen.
  - Magic Quotes Modus vorher abfragen.

Bereinigen mit und ohne Magic Quotes

```
<?php
   // Parameter bereinigen
   if ( get_magic_quotes_gpc() == 1 ) {
      $username = $_REQUEST['username'];
      $password = $_REQUEST['password'];
   }
   else {
      $username = mysqli_real_escape_string( $_REQUEST['username'] );
      $password = mysqli_real_escape_string( $_REQUEST['password'] );
   }
   // SQL String zusammenbauen
   $sql  = "SELECT ID,username,password FROM users ";
   $sql .= "WHERE username = '$username'";
   // ...
?>
```

Binden von Variablen in SQL Queries

Binden von Variablen in SQL Queries

- Man kann Platzhalter an Variablen binden.

Binden von Variablen in SQL Queries

- Man kann Platzhalter an Variablen binden.
  - möglich in der Datenliste `VALUES()`
  - möglich als Argument für `WHERE`

Binden von Variablen in SQL Queries

- Man kann Platzhalter an Variablen binden.
  - möglich in der Datenliste `VALUES()`
  - möglich als Argument für `WHERE`
- `mysqli_prepare()` bereitet SQL Statement vor.

Binden von Variablen in SQL Queries

- Man kann Platzhalter an Variablen binden.
  - möglich in der Datenliste `VALUES()`
  - möglich als Argument für `WHERE`
- `mysqli_prepare()` bereitet SQL Statement vor.
- `mysqli_stmt_bind_param()` bindet Parameter an Variablen.
- `mysqli_stmt_execute()` führt Statement aus.
- `mysqli_stmt_bind_result()` bindet Ergebnisse an Variablen.
- `mysqli_stmt_fetch()` holt Daten.

Binden von Variablen in SQL Queries (2)

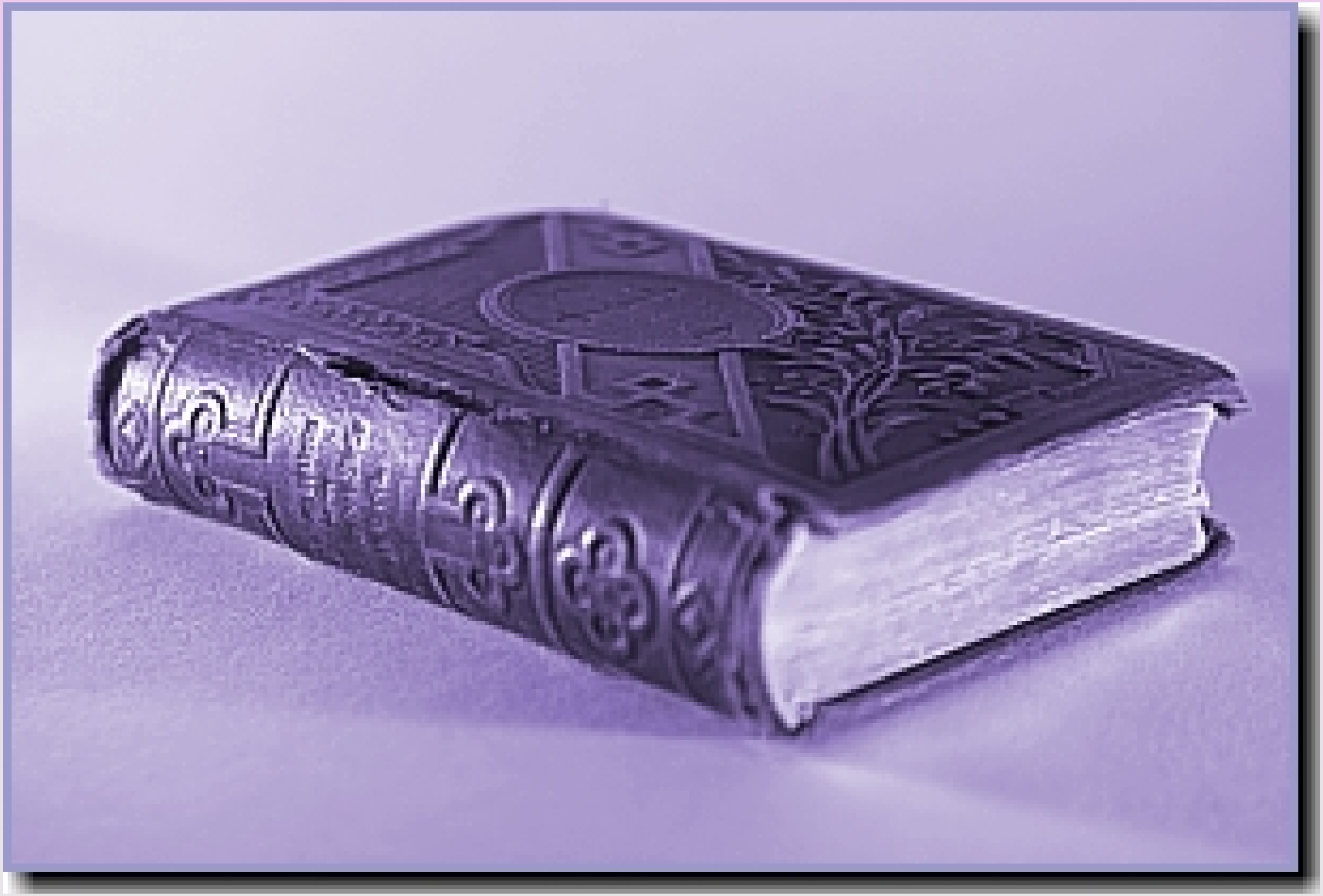
```
<?php
$stmt = mysqli_prepare($link,
         "INSERT INTO CountryLanguage VALUES (?, ?, ?, ?)");
mysqli_stmt_bind_param($stmt, 'sssd',
         $code, $language, $official, $percent);

$code = 'DEU';
$language = 'Bavarian';
$official = "F";
$percent = 11.2;

/* execute prepared statement */
mysqli_stmt_execute($stmt);

printf("%d Row inserted.\n", mysqli_stmt_affected_rows($stmt));
?>
```

Nützliche Programmiertechniken



Richtiges Speichern von Paßworten

Richtiges Speichern von Paßworten

- Paßworte niemals im Klartext speichern!

Richtiges Speichern von Paßworten

- Paßworte niemals im Klartext speichern!
- Verwenden von *Hash Funktionen*
  - `md5()` oder `sha1()` in PHP
  - `MD5()` oder `SHA1()` in MySQL

  Zu jedem Paßwort gibt es einen eindeutigen Hash Wert

Richtiges Speichern von Paßworten

- Paßworte niemals im Klartext speichern!
- Verwenden von *Hash Funktionen*
  - `md5()` oder `sha1()` in PHP
  - `MD5()` oder `SHA1()` in MySQL

  Zu jedem Paßwort gibt es einen eindeutigen Hash Wert
- Vergleichen des Hash Wertes

Dateiuploads

Dateiuploads

- Dateiupload geschieht durch Webformular.

Dateiuploads

- Dateiupload geschieht durch Webformular.
- Formular muß `enctype="multipart/form-data"` enthalten.

Dateiuploads

- Dateiupload geschieht durch Webformular.
- Formular muß `enctype="multipart/form-data"` enthalten.
- Datei wird vom PHP Interpreter temporär zwischengespeichert.

Dateiuploads

- Dateiupload geschieht durch Webformular.
- Formular muß `enctype="multipart/form-data"` enthalten.
- Datei wird vom PHP Interpreter temporär zwischengespeichert.
- PHP Skript bekommt Dateiinformationen in Superglobal `$_FILES`.

Dateiuploads

- Dateiupload geschieht durch Webformular.
- Formular muß `enctype="multipart/form-data"` enthalten.
- Datei wird vom PHP Interpreter temporär zwischengespeichert.
- PHP Skript bekommt Dateiinformationen in Superglobal `$_FILES`.
- PHP Skript muß Datei an Bestimmungsort kopieren.

Beispiel: Dateiuploads

Beispiel: Dateiuploads

```
...
<form enctype="multipart/form-data" action="__URL__" method="POST">
    <!-- MAX_FILE_SIZE must precede the file input field -->
    <input type="hidden" name="MAX_FILE_SIZE" value="30000" />
    <!-- Name of input element determines name in $_FILES array -->
    Send this file: <input name="userfile" type="file" />
    <input type="submit" value="Send File" />
</form>
...
```

`MAX_FILE_SIZE` ist nur eine *Empfehlung* an den Webbrowser!

Beispiel: Dateiuploads (2)

Beispiel: Dateiuploads (2)

- Dateiinformationen für *userfile* Input Tag:
  - `$_FILES['userfile']['name']` - Originaldateiname
  - `$_FILES['userfile']['type']` - Dateiart als MIME Type
  - `$_FILES['userfile']['size']` - Größe der Datei in Bytes
  - `$_FILES['userfile']['tmp_name']` - temporärer Dateiname
  - `$_FILES['userfile']['error']` - Fehlercode des Uploads
- Datei muß durch `move_uploaded_file()` gesichert werden.
- Entfernen von Pfadinformationen mittels `basename()`.
- PHP löscht temporäre Datei nach Ablauf des Skripts.

Dateidownloads

Dateidownloads

- Dateidownloads klingen simpel.

Dateidownloads

- Dateidownloads klingen simpel.
  - PHP Skript liest eine Datei und
  - schickt sie direkt zum Browser weiter.

`readfile()` ist nützlich dafür.

Dateidownloads

- Dateidownloads klingen simpel.
  - PHP Skript liest eine Datei und
  - schickt sie direkt zum Browser weiter.

  `readfile()` ist nützlich dafür.
- Skript muß Dateityp ankündigen.
  - `header()` Funktion schickt Beschreibung im HTTP Header:
  - `Content-disposition: attachment; filename=m.mpg`
  - `Content-type: video/mpeg`
  - `Content-Transfer-Encoding: Binary`
  - `Content-length: 14540439`

  `Content-type` gibt *MIME Typ* an

Dateidownloads

- Dateidownloads klingen simpel.
  - PHP Skript liest eine Datei und
  - schickt sie direkt zum Browser weiter.

  `readfile()` ist nützlich dafür.
- Skript muß Dateityp ankündigen.
  - `header()` Funktion schickt Beschreibung im HTTP Header:
  - `Content-disposition: attachment; filename=m.mpg`
  - `Content-type: video/mpeg`
  - `Content-Transfer-Encoding: Binary`
  - `Content-length: 14540439`

  `Content-type` gibt *MIME Typ* an
- Notwendig, damit Browser „automatisch“ Aktionen durchführt.

Beispiel: Dateidownloads

Beispiel: Dateidownloads

```
<?
$dir="/path/to/file/";
if (isset($_REQUEST["file"])) {
    $file=$dir.$_REQUEST["file"];
    header("Content-type: application/force-download");
    header("Content-Transfer-Encoding: Binary");
    header("Content-length: ".filesize($file));
    header("Content-disposition: attachment; filename=\"".
           basename($file)."\"");
    readfile("$file");
} else {
    echo "No file selected";
}
?>
```

Sessions

Sessions

- Sessions ordnen Seitenaufrufe bestimmten Clients zu.

Sessions

- Sessions ordnen Seitenaufrufe bestimmten Clients zu.
  - Der Client speichert eine *Session-ID*.
  - Session-IDs werden als Cookie oder Parameter übertragen.
  - Der Server speichert die zur ID zugeordneten Daten.

Sessions

- Sessions ordnen Seitenaufrufe bestimmten Clients zu.
  - Der Client speichert eine *Session-ID*.
  - Session-IDs werden als Cookie oder Parameter übertragen.
  - Der Server speichert die zur ID zugeordneten Daten.
- Sessions ermöglichen Zustände in HTTP Übertragungen.

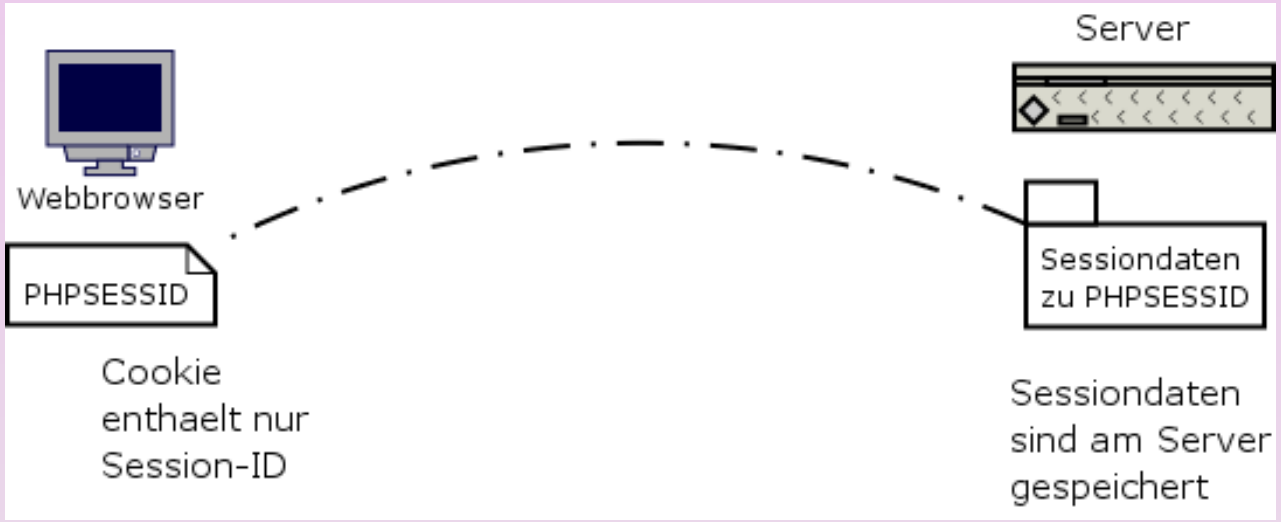
Sessions

- Sessions ordnen Seitenaufrufe bestimmten Clients zu.
  - Der Client speichert eine *Session-ID*.
  - Session-IDs werden als Cookie oder Parameter übertragen.
  - Der Server speichert die zur ID zugeordneten Daten.
- Sessions ermöglichen Zustände in HTTP Übertragungen.
  - Status *logged in/out*.
  - Benutzerdefinierte Einstellungen.

Sessions

- Sessions ordnen Seitenaufrufe bestimmten Clients zu.
  - Der Client speichert eine *Session-ID*.
  - Session-IDs werden als Cookie oder Parameter übertragen.
  - Der Server speichert die zur ID zugeordneten Daten.
- Sessions ermöglichen Zustände in HTTP Übertragungen.
  - Status *logged in/out*.
  - Benutzerdefinierte Einstellungen.
- PHP verwaltet Sessions automatisch.

Überblick Sessionverwaltung



PHP Funktionen zu Sessions

PHP Funktionen zu Sessions

- `session_start()` startet eine Session.

PHP Funktionen zu Sessions

- `session_start()` startet eine Session.
  - `session_start()` muß wie `header()` am Anfang stehen!

PHP Funktionen zu Sessions

- `session_start()` startet eine Session.
  - `session_start()` muß wie `header()` am Anfang stehen!
- Sessiondaten schreibt man in den Superglobal `$_SESSION[]`.
  - `$_SESSION['logged_in']=1;`
  - `$_SESSION['username']='alf'`

PHP Funktionen zu Sessions

- `session_start()` startet eine Session.
  - `session_start()` muß wie `header()` am Anfang stehen!
- Sessiondaten schreibt man in den Superglobal `$_SESSION[]`.
  - `$_SESSION['logged_in']=1;`
  - `$_SESSION['username']='alf'`
- `session_destroy()` löscht alle Daten einer Session.

PHP Funktionen zu Sessions

- `session_start()` startet eine Session.
  - `session_start()` muß wie `header()` am Anfang stehen!
- Sessiondaten schreibt man in den Superglobal `$_SESSION[]`.
  - `$_SESSION['logged_in']=1;`
  - `$_SESSION['username']='alf'`
- `session_destroy()` löscht alle Daten einer Session.
- `session_regenerate_id()` erzeugt eine neue Session-ID.

Wichtige Eigenschaften von Sessions

Wichtige Eigenschaften von Sessions

- Session-ID beschreibt Session.

Wichtige Eigenschaften von Sessions

- Session-ID beschreibt Session.
  - Senden richtiger Session-ID übernimmt Session!
  - Bei kritischen Daten immer Verschlüsselung (HTTPS) verwenden!

Wichtige Eigenschaften von Sessions

- Session-ID beschreibt Session.
  - Senden richtiger Session-ID übernimmt Session!
  - Bei kritischen Daten immer Verschlüsselung (HTTPS) verwenden!
- Session-ID liegt in Cookies am Client.

Wichtige Eigenschaften von Sessions

- Session-ID beschreibt Session.
  - Senden richtiger Session-ID übernimmt Session!
  - Bei kritischen Daten immer Verschlüsselung (HTTPS) verwenden!
- Session-ID liegt in Cookies am Client.
  - Sessions richtig terminieren.
  - Sicherheit am Client spielt auch eine Rolle.

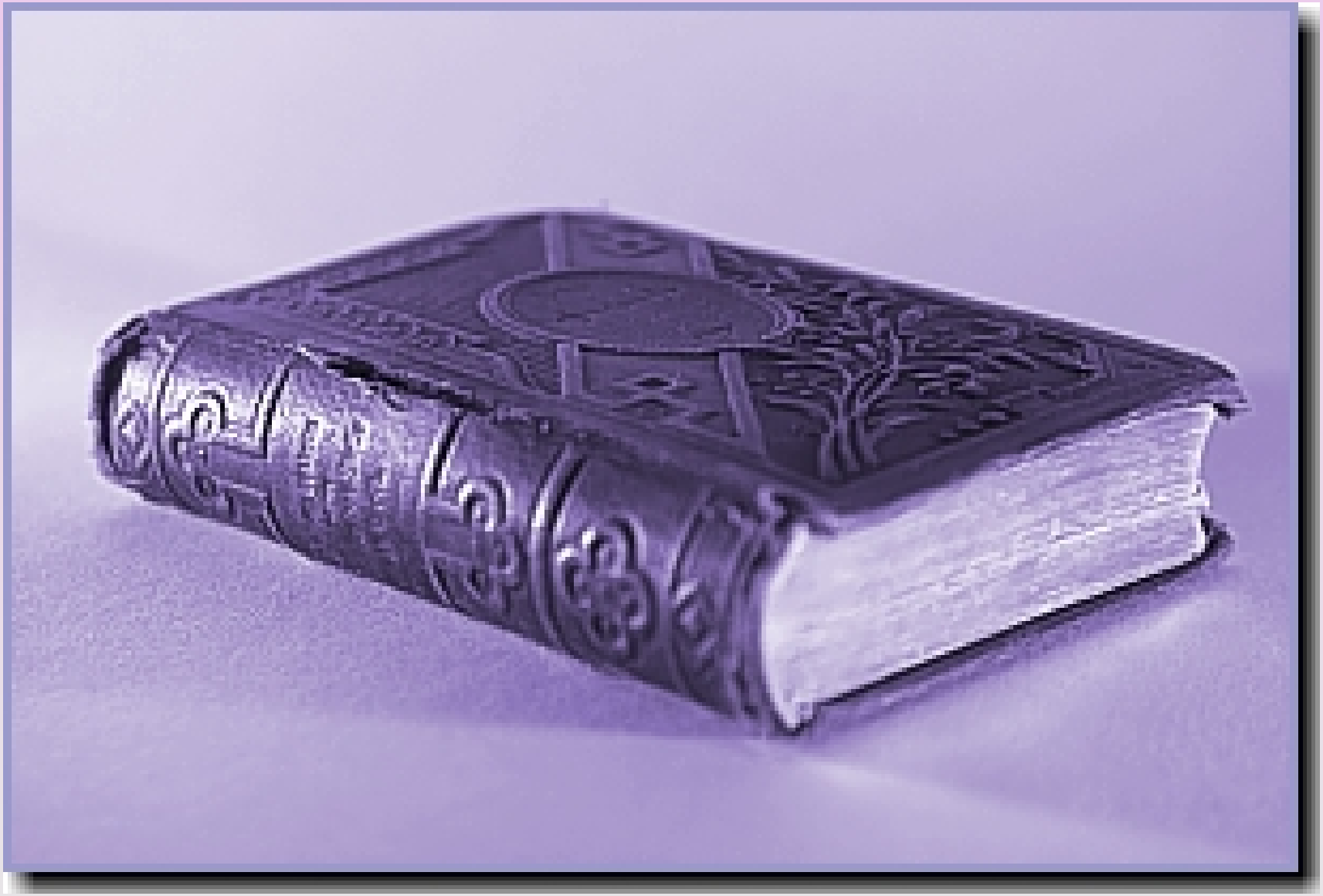
Wichtige Eigenschaften von Sessions

- Session-ID beschreibt Session.
  - Senden richtiger Session-ID übernimmt Session!
  - Bei kritischen Daten immer Verschlüsselung (HTTPS) verwenden!
- Session-ID liegt in Cookies am Client.
  - Sessions richtig terminieren.
  - Sicherheit am Client spielt auch eine Rolle.
- Sessiondaten liegen am Server.

Wichtige Eigenschaften von Sessions

- Session-ID beschreibt Session.
  - Senden richtiger Session-ID übernimmt Session!
  - Bei kritischen Daten immer Verschlüsselung (HTTPS) verwenden!
- Session-ID liegt in Cookies am Client.
  - Sessions richtig terminieren.
  - Sicherheit am Client spielt auch eine Rolle.
- Sessiondaten liegen am Server.
  - Alle PHP Skripte haben potentiellen Zugriff.
  - Andere Benutzer können prinzipiell Zugriff erlangen.
    - ★ Uneingeschränkter Upload ist dazu sehr hilfreich.

Programmierbeispiele



Beispiel: Taschenrechner

- zwei Eingabefelder für Zahlen
- Selectbox für Rechenoperationen
- PHP Skript errechnet Ergebnis
- Darstellung wieder im Formular
- Optional: PHP läßt MySQL rechnen

Beispiel: Gästebuch

Wir brauchen ein Gästebuch mit

- Name
- Ort
- E-Mail-Adresse
- Homepage
- Titel des Eintrags
- Textfeld mit Eintrag

Beispiel: Gästebuch - mögliche SQL Struktur

```
CREATE TABLE guestbook (
    ID INT AUTO_INCREMENT,
    Name VARCHAR(128),
    Ort VARCHAR(128),
    Email VARCHAR(160),
    Homepage VARCHAR(240),
    Titel VARCHAR(240),
    Beitrag TEXT,
    PRIMARY KEY ('ID')
);
```

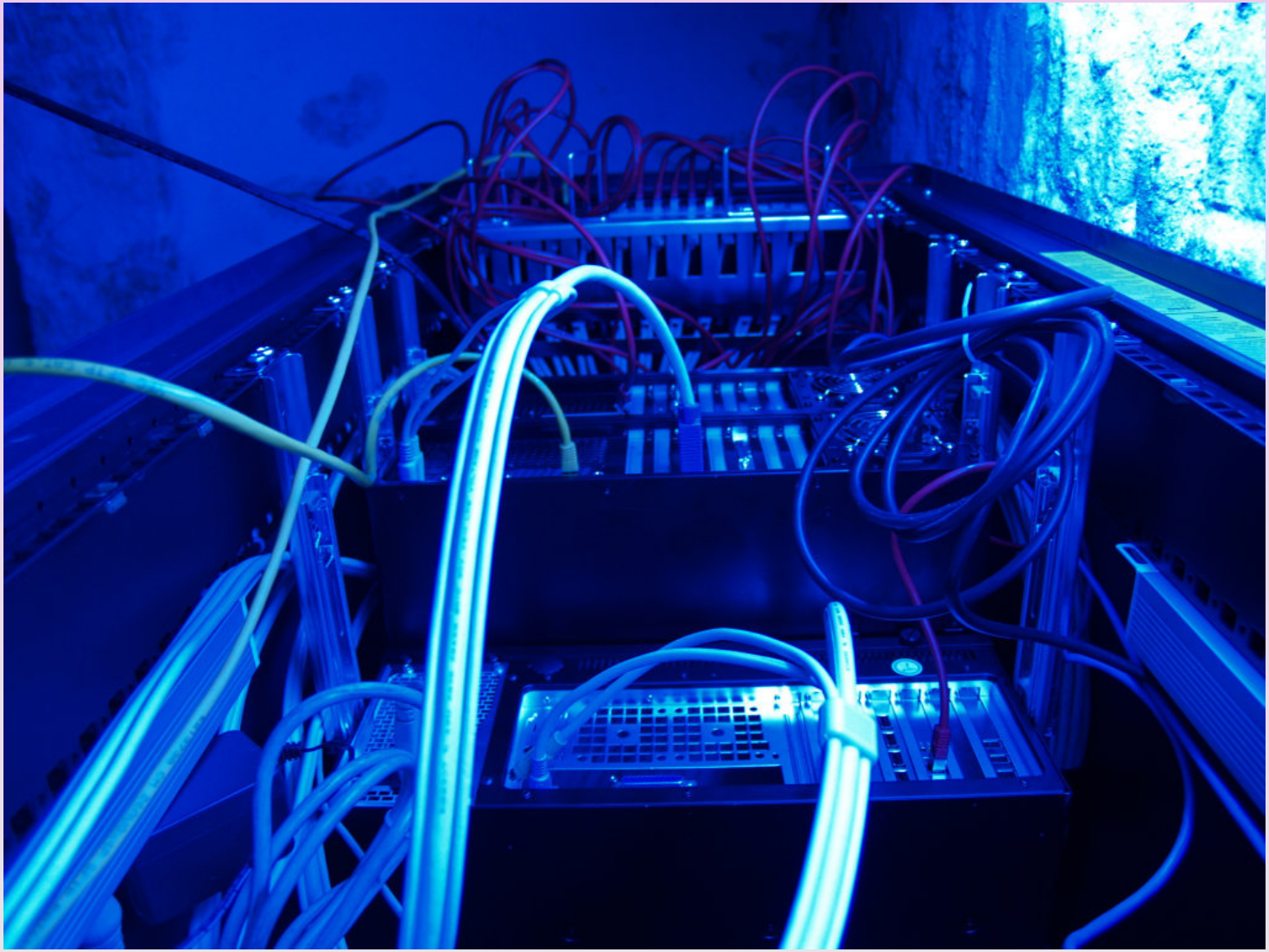
Beispiel: Webblog

- Datenbank mit Tabelle für Einträge
  - Text des Blogeintrags
  - Autor
  - Titel
  - Erstellungszeit und -datum
- Login und Logout für Autoren
- Skript zum Eingeben und Editieren
- Skript zur Anzeige aller Blogeinträge

Beispiel: RSS Feeds generieren

- Programmierung eines kleinen Blogs:
  - `index.php` zum Anzeigen der Startseite
  - `eintrag.php` zum Anzeigen eines Eintrags
  - `edit.php` zum Eingeben oder Bearbeiten eines Eintrags
  - `delete.php` zum Löschen eines Eintrags
- Export aller Einträge mittels RSS
  - `rss.php` generiert RSS Ausgabe
  - Ausgabe in den Formaten
    - RSS 2.0
    - Atom
- Session Handling für alle Verwaltungsskripte

So long and thanks for all the fish...



Referenzen

- Andreas Buchmann & Ralf Smolarek, *PHP interaktiv*, 2. Auflage, ISBN 3-936121-01-X, Omnigena, 2006.
- Kevin Tatroe, Rasmus Lerdorf & Peter MacIntyre, *Programming PHP*, O'Reilly & Associates, Inc., 2nd Edition, April 2006.
- PHP Freaks Tutorial Section
- PHP Online Manual
- SelfPHP
- Zend Developer Zone

Über dieses Dokument

- Autor: René Pfeiffer
- Erstellt mit LaTeX und LaTeX Beamer Class
- Dokumentensammlung unter

  http://web.luchs.at/information/docs.php