Webengineering

PHP

PHP - Historie

1994	Rasmus Lerdorf: Entwickelt eine kleine von C abgeleitete Skriptmaschine. Ziel: Programmierung eines Web-Servers Name: Personal Home Page Tools (PHP) Quellcode wird im Internet freigegeben ⟶ PHP Version 1
1995	Open-Source-Szene entwickelt PHP weiter zu PHP/FI (FI = From Interface) ⟶ PHP Version 2 (.phtml)
1997	PHP-Version 3 wird veröffentlicht (.php3)
2000	PHP-Version 4 wird veröffentlicht (.php4) Neuer Kernel ZEND (von Zeev Suraski und Andi Gutman)
2004	PHP-Version 5 wird veröffentlicht (.php5) ⟶ ZEND Engine II

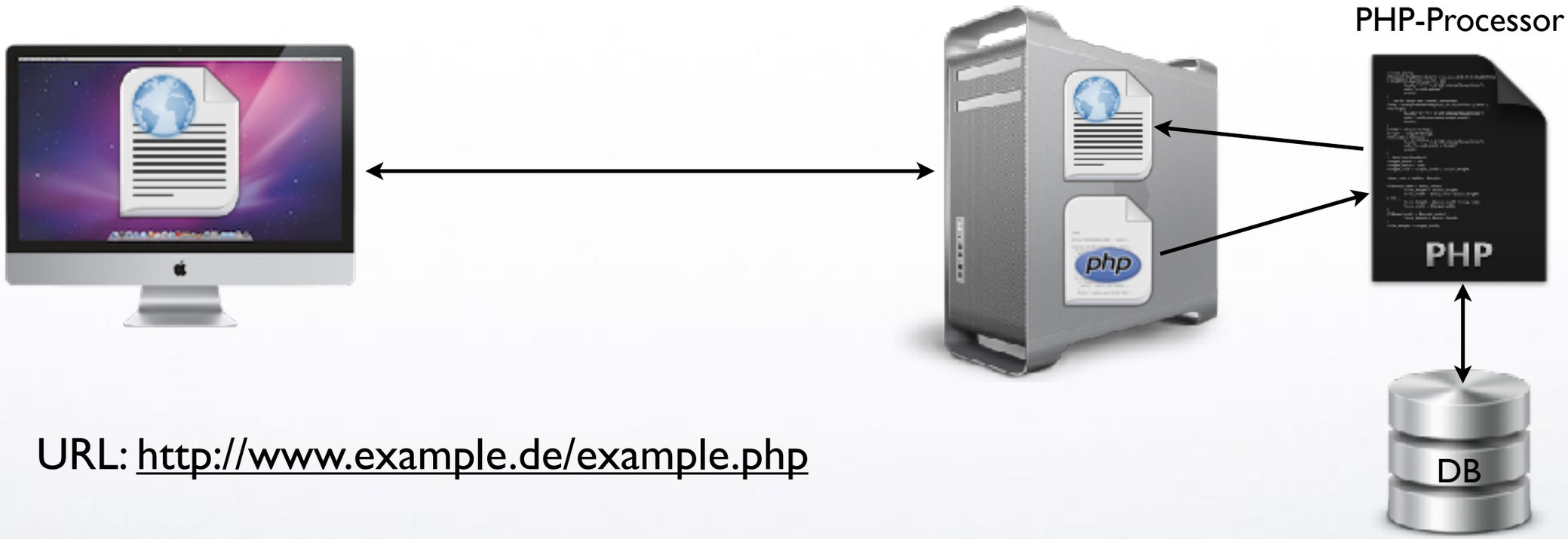
Alternativen:

Perl 1986 von Larry Wall für die NSA entwickelt, sehr gute String-Verarbeitung

ASP Active Server Pages, basiert auf VBScript (nur Microsoft)

Python, Java, Tcl, ...

PHP - Funktionsweise



URL: http://www.example.de/example.php

- Dateiendung „.php“ signalisiert dem Webserver die Datei an den PHP-Prozessor zu schicken

PHP - ein Beispiel

```
<html>
 <head> </head>
 <body>
   <h2>SGML-Style</h2>

   <?
     echo 'Hello World!';
   ?>
 </body>
</html>
```

```
<html>
 <head> </head>
 <body>
   <h2>XML-Style</h2>

   <?php
     echo 'Hello World!';
   ?>
 </body>
</html>
```

```
<html>
 <head> </head>
 <body>
   <h2>ASP-Style</h2>

   <%
     echo 'Hello World!';
   %>
 </body>
</html>
```

- Einbindung von PHP-Anweisungen in ein HTML-Dokument geschieht ähnlich eines Kommentars
- Syntax an C angelehnt: Anweisungen werden mit Semikolon beendet
- ASP-Style muss in php.ini eingestellt werden

PHP - Kommentare

```php
<?php
  echo 'einzeiliger Kommentar.';
  //einzeiliger C-Kommentar bis Zeilenende oder PHP-Blockende
?>
```

```php
<?php
  echo 'Zeilenkommentar';
  #einzeiliger Shell-Kommentar bis Zeilenende oder PHP-Blockende
?>
```

```php
<?php
  echo 'Zeilenkommentar';
  /* mehrzeiliger Kommentar,
     der bis zum Kommentarende geht und auch
     '?>' und HTML einschließt.
  */
?>
```

PHP - Grundlagen

Ausgabe der Version und der Einstellungen (php.ini):

```
<?php
    phpinfo();          // echo impliziert
    echo phpversion(); // PHP-Version
?>
```

Zahlenliterale:

56.60	Dezimal, Gleitkommawert
45	Dezimal, Ganzzahlwert
0xFF	Hexadezimal
061	Octal

PHP - Grundlagen

Dateien einschließen:

```
<?php
    include "inc/kopf.inc.php";
    require ("inc/kopf2.inc.php");
    include_once ("../db.php");
    require_once "../functions.php";
?>
```

- Relative und absolute Pfadangaben möglich
- Sind Befehle (keine Funktionen) daher können Klammern auch entfallen
- Bei fehlender Datei: Warnung (bei include), Fehler (bei require)
- Bei require wird Datei erst eingebunden und Dokument im Anschluss geparst
- Bei include wird die Datei erst eingebunden, wenn der Parser am Statement angekommen ist
- Mit „_once“ wird die Datei nur einmal eingebunden (Fehlervermeidung)

PHP - Grundlagen

Ausgabe von HTML:

```
<?php
    echo "<p>Dieser Text wird angezeigt.</p>";
    echo '<p>Das Buch kostet $59 bzw. ', $value, ' Euro.</p>';
    print ("<h1>Es ist jetzt {$zeit[1]} Uhr.</h1>");
    print '<div>Print geht auch ohne Klammern!</div>';
?>
<?=$ausgabe?> <!-- Kurzschreibweise zur Ausgabe einer Variablen -->
```

- "echo" ist ein Befehl, "print" ist eine Funktion (hat Rückgabewert)
- "echo" erwartet beliebig viele Argumente, "print" genau ein Argument
- String mit ' ' interpretiert keine maskierten Zeichen oder Variablen
- String mit " " interpretiert maskierte Zeichen und Variablen (wertet diese aus)
- Zur Vermeidung der Fehlinterpretation einer Variablen im String mit " ", kann die Variable auch innerhalb von geschweiften Klammern stehen

PHP - Grundlagen

Maskierte Zeichen:

`\' \"`	Anführungszeichen	`\$`	Dollarzeichen
`\n`	Zeilenumbruch (newline)	`\{ \}`	geschweifte Klammern
`\r`	Zeilenumbruch (carriage return)	`\[ \]`	Eckige Klammern
`\t`	Tabulator	`\0-\777`	Zeichen in Octal
`\\`	Backslash	`\0x-\xFF`	Zeichen in Hexadecimal

Zeilenumbrüche:

`\r\n`	Zeilenumbruch Windows	`\n`	Zeilenumbruch Unix
		`\r`	Zeilenumbruch Mac

PHP - Grundlagen

Variablen:

```php
<?php
    $name = "Michael Krause";
    $i    = 1;
    $zahl = 14.59;
?>
```

- Beginnen immer mit $
- Erlaubte Zeichen: A-Z, a-z, 0-9 und _
- Variablenname muss mit Buchstaben oder _ beginnen
- Zwischen Groß- und Kleinschreibung wird unterschieden
- Variablentyp wird bei der Zuweisung automatisch erkannt und verwaltet

PHP - Grundlagen

Gültigkeitsbereich:

```
<?php
    $gzahl = 22; //globale Variable
    function ausgabe() {
        $lzahl = 44;    //lokale Variable
        echo $gzahl;    //Ausgabe: <nichts>
        echo $lzahl;    //Ausgabe: 44
        global $gzahl;
        echo $gzahl;    //Ausgabe: 22
    }
    ausgabe();
?>
```

- Variablen sind nur innerhalb ihres definierten Blocks/Bereichs gültig
- Lokale Variablen "überschreiben" globale Variablen
- Mit dem Befehl "global" kann innerhalb von Funktionen auf globale Variablen zugegriffen werden

PHP - Grundlagen

Statische Variablen:

```php
<?php
    function ausgabe() {
        static $zahl = 20;
        echo "$zahl<br>\n";
        $zahl++
    }
    ausgabe(); //Ausgabe: 20
    ausgabe(); //Ausgabe: 21
    ausgabe(); //Ausgabe: 22
?>
```

- Statische Variablen bleiben auch nach dem Verlassen einer Funktion erhalten
- Wichtigste Anwendung ist in rekursiven Funktionen

PHP - Grundlagen

Dynamische Variablen:

```
<?php
    $dynvar  = "name";
    $$dynvar = "PHP";
    echo $name;  //Ausgabe: PHP
?>
```

- Wert der Variablen wird als Name der dynamischen Variablen genommen
- Dynamische Variablen werden mit $$ angesprochen
- Einmal erfolgte Zuweisungen bleiben von späteren Umbenennungen der früheren Variablen unberührt.

PHP - Grundlagen

Referenzen:

```
<?php
    $zahl  = 14;
    $ziel = &$zahl;
    $zahl++;
    echo $ziel; //Ausgabe: 15
?>
```

- Referenzen sind Verweise auf eine andere Variable zeigen
- Wert wird nicht kopiert, sondern bleibt an der ursprünglichen Stelle definiert
- Änderungen der Quelle wirken sich auf die Referenzen aus

PHP - Grundlagen

Konstanten:

```php
<?php
    define("PI", 3.14159265);
    echo defined($PI);  //Ausgabe: 3.14159265
    print_r(get_defined_constants()); //Ausgabe aller Konstanten
?>
```

- Konvention: Name in Großbuchstaben
- Können nicht mehr nachträglich (zur Laufzeit) geändert werden
- defined($const): Ermittelt, ob eine Konstante definiert wurde
- get_defined_constants(): Erzeugt ein Array mit allen definierten Konstanten

PHP - Grundlagen

Vordefinierte Konstanten:

DEFAULT_INCLUDE_PATH	Standardpfad für Dateisuche
PEAR_INSTALL_DIR	Installationsordner der Erweiterung PEAR
PEAR_EXTENSION_DIR	Erweiterungsordner der Erweiterung PEAR
PHP_EXTENSION_DIR	Pfad zu kompilierten Erweiterungen
PHP_BINDIR	Verzeichnis zu PHP selbst
PHP_LIBDIR	Verzeichnis der Bibliotheken
PHP_DATADIR	Datenverzeichnis
PHP_SYSCONFDIR	Konfigurationsverzeichnis
PHP_LOCALSTATEDIR	Ablageort lokaler Statusdateien
PHP_CONFIG_FILE_PATH	Pfad zur *php.ini*

PHP - Grundlagen

Debugging von Variablen

```php
<?php
    $a = array("first", 508, TRUE);
    var_dump($a);
    var_export($a);
    print_r($a);
?>
```

var_dump	Gibt Informationen über Variablenstruktur und Inhalt
var_export	Gibt Informationen über Variablenstruktur und Inhalt, erzeugt aber auch validen PHP-Code um den Inhalt wiederherzustellen
print_r	Gibt Informationen über Variablenstruktur und Inhalt in gut lesbarer Form

PHP - Grundlagen

Vordefinierte Konstanten:

TRUE (true)	"Wahr" (intern: nummerischer Wert ungleich 0)
FALSE (false)	"Falsch" (intern: nummerischer Wert 0)
NULL (null)	"Nichts" (nicht vorhandener Wert)

Pseudokonstanten:

__FILE__	Enthält den Dateinamen des Skripts
__LINE__	Enthält die Zeilennummer in der sich der Befehl befindet
__CLASS__	Enthält die aktuelle Klasse
__METHOD__	Enthält die aktuelle Methode innerhalb der Klasse
__FUNCTION__	Enthält den Namen der Funktion

PHP - Grundlagen

Datentypen:

integer, int	Ganzzahl
boolean, bool	Logischer Wert: 0 (FALSE), 1 (TRUE)
double, real	Fließkommazahl
string	Zeichenkette (0-2.000.000.000 Zeichen)
resource	Zeiger auf externe Datenquelle (Datei, DB)
array	Array
object	Objekt
null	NULL (leere Variable ohne speziellen Typ)

PHP - Grundlagen

Konvertierung von Datentypen:

```
<?php
    $zahl   = 1;
    $double = (double) $zahl; //Ausgabe: 1.0
    $int    = intval($double); //Ausgabe: 1
    $bool   = settype($int, "bool"); //Ausgabe: TRUE
?>
```

- Konvertierung auf 3 unterschiedliche Weisen:
  - Cast: `([type]) $variable;`
  - Funktion settype: `settype($variable, "[type]");`
  - val-Funktion: `[type]val($variable);`

PHP - Grundlagen

Variablentypen und -zustände:

```
<?php
    echo isset($zahl);    //Ausgabe: 0
    $zahl = 0;
    echo gettype($zahl); //Ausgabe: integer
    echo empty($zahl);    //Ausgabe: 1
    unset($zahl);
    echo is_null($zahl); //Ausgabe: 1
?>
```

- isset: Prüft, ob eine Variable existiert
- unset: Hebt die Zuweisung einer Variablen auf
- empty: Prüft ob eine existierende Variable 0 oder "" enthält
- Bestimmung des Datentyps auf 2 unterschiedliche Weisen:
  - Funktion gettype: `gettype($variable)`
  - is-Funktionen: `is_[type]($variable)`

PHP - Grundlagen

Arithmetische Operatoren:

```php
<?php
    $x + $y;   // Addition
    $x - $y;   // Subtraktion
    $x * $y;   // Multiplikation
    $x / $y;   // Division
    $x % $y;   // Modulus (Rest der Ganzzahldivision)
    $x++;      // Inkrementation
    $x--;      // Dekrementation
?>
```

Logische Operatoren:

```php
<?php
    $x and $y; // wahr: $x UND $y sind wahr
    $x or $y;  // wahr: $x ODER $y sind wahr
    $x xor $y; // wahr: $x UND $y ungleich
    $x && $y;  // entspricht and
    $x || $y;  // entspricht or
    !$x;       // wahr: $x ist falsch (negierter Wert)
?>
```

PHP - Grundlagen

Vergleichsoperatoren:

```php
<?php
    $x == $y;   // Gleichheit
    $x != $y;   // Ungleichheit
    $x === $y;  // Identisch
    $x !== $y;  // Nicht identisch
    $x < $y;    // Größer als
    $x > $y;    // Kleiner als
    $x <= $y;   // Größer als oder gleich
    $x >= $y;   // Kleiner als oder gleich
?>
```

PHP - Grundlagen

Zuweisungsoperatoren:

```php
<?php
    $x = $y;    // Zuweisung
    $x += $y;   // Hinzuaddieren einer Zahl zu $x
    $x -= $y;   // Abzug einer Zahl von $x
    $x *= $y;   // Vielfaches von $x
    $x /= $y;   // Teilung von $x
    $x %= $y;   // $x % $y und Zuweisung des Ergebnisses in $x
    $s .= $p;   // Erweiterung einer Zeichenkette
?>
```

Fehleroperator:

```php
<?php
    @require "not/existing/file.php";
?>
```

- Unterdrückt Fehlermeldungen und lässt Skript weiterlaufen
- Unterdrückt nur Fehler in Laufzeit (keine Parser-Fehler)

PHP - Grundlagen

Stringmanipulation:

```
<?php trim(string s); ?>
```

- Entfernt Leerzeichen am Anfang und Ende eines Strings

```
<?php strlen(string s); ?>
```

- Ermittelt die Anzahl der Zeichen einer Zeichenkette

```
<?php strrev(string s); ?>
```

- Kehrt die Reihenfolge der Zeichen einer Zeichenkette um

```
<?php str_replace(mixed a, mixed b, mixed s, [integer c]); ?>
```

- Ersetzt im String s (falls s ein Array: allen Strings in s) die nächsten c Vorkommen aller a mit denen aus b

PHP - Grundlagen

Stringmanipulation:

```
<?php strtolower(string s); ?>
```

- Wandelt alle Großbuchstaben in Kleinbuchstaben um

```
<?php strtoupper(string s); ?>
```

- Wandelt alle Kleinbuchstaben in Großbuchstaben um

```
<?php explode(string sep, string s, [integer c]); ?>
```

- Zerlegt den String in ein Array an den nächsten c Vorkommen des Seperators

```
<?php implode(string sep, array a); ?>
```

- Setzt eine Zeichenkette aus einem Array zusammen und fügt den angegebenen Separator zwischen den einzelnen Teilen ein

PHP - Grundlagen

Stringmanipulation:

```
<?php strpos(string s, mixed n, [integer o]); ?>
```

- Bestimmt die Position des ersten Vorkommens von n in s, beginnend bei o

```
<?php strrpos(string s, string n, [integer o]); ?>
```

- Bestimmt die Position des letzten Vorkommens von n in s, beginnend bei o

```
<?php strrchr(string s, string n); ?>
```

- Gibt den String ab dem letzten Vorkommen von n zurück

```
<?php substr(string s, integer o, [integer l]); ?>
```

- Gibt den Teil-String mit der Länge l ab o zurück

PHP - Grundlagen

Stringmanipulation:

```
<?php strstr(string s, mixed n, [integer o]); ?>
```

- Gibt den Teil-String ab der Position von n in s (beginnend bei o) zurück

```
<?php strspn(string s, string m, [integer o]); ?>
```

- Gibt den Anfang von s zurück, der nur Zeichen aus m enthält (max. Länge l)

```
<?php chr(integer i); ?>
```

- Gibt das Zeichen zu einem ASCII-Wert zurück

```
<?php ord(string s); ?>
```

- Gibt den ASCII-Wert zu einem Zeichen zurück

PHP - Grundlagen

Stringmanipulation:

```
<?php quotemeta(string s); ?>
```

- Backslash vor jedem Vorkommen folgender Zeichen: . \ + * ? [^] ($)

```
<?php addslashes(string s); ?>
```

- Backslash vor wichtigen Zeichen in DB-Anfragen: ' " \ NUL

```
<?php stripslashes(string s); ?>
```

- Macht das Gegenteil von addslashes()

```
<?php strcmp(string s1, string s2); ?>
```

- Vergleicht zwei Zeichenketten

  Rückgabewert: -1 (links kleiner), 0 (gleich), +1 (rechts kleiner)

PHP - Grundlagen

HTML-Funktionen:

```
<?php htmlspecialchars(string s); ?>
```

- Wandelt folgende Zeichen in HTML-Entities um: & ' " < >

```
<?php htmlentities(string s); ?>
```

- Wandelt alle Zeichen, die als HTML-Entity geschrieben werden können, um

```
<?php nl2br(string s, [bool is_xml]); ?>
```

- Wandelt alle Zeilenumbrüche in
-Tags um

```
<?php [raw]urlencode(string s); ?>
```

- Alle [auch unerlaubte] Zeichen werden URL-kodiert (%-Schreibweise)
- Pendant: [raw]urldecode($s)

PHP - Grundlagen

Weitere Funktionen:

- Mathematische Funktionen
  - floor, ceil, round
  - cos, exp, pow
  - min, max
  - is_nan
- Datums- und Zeitfunktionen
  - date
  - mktime
  - strftime

PHP - Grundlagen

Arrays:

```php
<?php
    arr[] = "first"; //arr[0] = "first"
    arr[] = 508;     //arr[1] = 508
    arr[] = TRUE;    //arr[2] = TRUE

    arr = array("first", 508, TRUE);

    arr2[3]  = "first";
    arr2[]   = 508;  //arr2[4] = 508
    arr2[10] = TRUE; //arr2[5] = NULL
?>
```

indiziertes Array

```php
<?php
    arr1["Val1"] = "first";
    arr1["Val2"] = 508;
    arr1["Val3"] = TRUE;

    arr2 = array("Val1" => "first",
                 "Val2" => 508,
                 "Val3" => TRUE);

    arr = arr1 + arr2;
?>
```

assoziatives Array

- „+“-Operator kombiniert 2 Arrays

PHP - Grundlagen

Array-Funktionen:

```
<?php count(mixed a); ?>
```

- Gibt die Anzahl der Elemente eines Arrays zurück

```
<?php in_array(mixed n, array a); ?>
```

- Prüft ob ein Elements aus n als Wert in Array a vorhanden ist

```
<?php array_pop(array a); ?>
```

- Entfernt das letzte Element und gibt dieses zurück

```
<?php array_push(array a, mixed v, ...); ?>
```

- Fügt mehrere Elemente ans Ende des Arrays an

PHP - Grundlagen

Array-Funktionen:

```
<?php array_merge(array a1, array a2, ...); ?>
```

- Verbindet zwei oder mehrere Arrays miteinander

```
<?php array_reverse(array a); ?>
```

- Dreht das Array komplett um (erstes Element wird letztes, usw.)

```
<?php array_unique(array a); ?>
```

- Entfernt doppelte Einträge im Array

PHP - Grundlagen

Array-Funktionen:

```
<?php sort(array a); ?>
```

- Sortiert ein eindimensionales Array vorwärts

```
<?php rsort(array a); ?>
```

- Sortiert ein eindimensionales Array rückwärts

```
<?php asort(array a); ?>
```

- Sortiert Array anhand der Werte, wobei die Indizes erhalten bleiben

```
<?php ksort(array a); ?>
```

- Sortiert Array anhand der Schlüssel (bei assoziativen Arrays sinnvoll)

PHP - Grundlagen

Array-Funktionen:

```
<?php array_flip(array a); ?>
```

- Vertauscht Schlüssel mit Werten

```
<?php array_key_exists(mixed k, array a); ?>
```

- Prüft, ob ein Schlüssel im Array existiert

```
<?php array_keys(array a); ?>
```

- Gibt alle Schlüssel als Array zurück (sinnvoll bei assoziativen Arrays)

```
<?php array_values(array a); ?>
```

- Gibt alle Werte als Array zurück (sinnvoll bei assoziativen Arrays)

PHP - Grundlagen

Superglobale Arrays:

$GLOBALS	Alle globalen Variablen
$_SERVER	Servervariablen
$_GET	GET-Parameter des letzten URL
$_POST	POST-Parameter des letzten Formulars
$_COOKIE	Vom Browser gesendete Cookies
$_FILES	Hochgeladene Dateien
$_ENV	Umgebungsvariablen des Servers
$_REQUEST	Alle Anforderungsvariablen zusammen (GET/POST)
$_SESSION	Sitzungsvariablen

PHP - Grundlagen

$_SERVER:

PHP_SELF	Dateiname des aktuell ausgeführten Skripts
SERVER_ADDR	Die IP-Adresse der Servers auf dem das Skript ausgeführt wird
SERVER_NAME	Hostname des Servers auf dem das Skript ausgeführt wird
REQUEST_METHOD	Verwendete Requestmethode zum Zugriff auf die Seite
QUERY_STRING	Der Querystring mit dem auf die Seite zugegriffen wurde
DOCUMENT_ROOT	Das Dokument-Rootverzeichnis aus der Serverkonfiguration
REMOTE_ADDR	IP-Adresse des Benutzers, der auf die Seite zugreift
REMOTE_HOST	Hostname des Benutzers, der auf die Seite zugreift
REMOTE_PORT	Port des Benutzers, der auf die Seite zugreift
SCRIPT_FILENAME	absoluter Pfad zum aktuell ausgeführten Skript

PHP - Grundlagen

Bedingungen - if/elseif/else:

```
<?php
  if ($test) {
    // Blockanweisung
  } elseif ($test2) {
    // Blockanweisung
  } else {
    // Blockanweisung
  }
?>
```

```
<?php if ($t) { ?>
    // HTML
<?php } elseif ($t2) { ?>
    // HTML
<?php } else { ?>
    // HTML
<?php } ?>
```

```
<?php if ($t): ?>
    // HTML
<?php elseif ($t2): ?>
    // HTML
<?php else: ?>
    // HTML
<?php endif ?>
```

Kurzschreibweise:

```
<?php
  if ($t) {
    $r = TRUE
  } else {
    $r = FALSE
  }
?>
```

```
<?php
  $r = $t ? TRUE : FALSE;
?>
```

PHP - Grundlagen

Bedingungen - switch/case:

```
<?php
    $stunde = date("H");
    switch($stunde) {
        case 8:
            echo "Guten Morgen";
            break;
        case 9:
            echo "Bisschen spät heute?";
            break;
        case 10:
            echo "Jetzt gibt‘s Ärger";
            break;
        case 11:
            echo "Lass dich krankschreiben";
            break;
        default:
            echo "Sonstwann am Tag ...";
    }
?>
```

PHP - Grundlagen

Schleifen:

```php
<?php
  while ($test) {
    // Blockanweisung
  }

  do {
    // Blockanweisung
  } while ($test)
?>
```

while-Schleife

```php
<?php
  for ($i=0; $i<5; $i++) {
    print $i;
  }

  for (;;) {
    if ($test) break;
  }
?>
```

for-Schleife

```php
<?php
  foreach ($a as $v) {
    echo $v;
  }

  foreach ($a as $k=>$v) {
    echo $k, " - ", $v;
  }
?>
```

foreach-Schleife

PHP - Grundlagen

Funktionen:

```php
<?php
    function name($param, $param2, ...) {
        // Blockanweisung
        return $value;
    }
?>
```

```php
<?php function name($param, $param2 = "optional") {} ?>
```

- Definition optionaler Parameter (müssen als letzte Parameter definiert werden)

```php
<?php function name($param, $argv) {} ?>
```

- Übergabe beliebig vieler Parameter (werden im Array argv zusammengefasst)

PHP - Grundlagen

Formulare:

$_GET	GET-Parameter des letzten URL
$_POST	POST-Parameter des letzten Formulars
$_REQUEST	Alle Anforderungsvariablen zusammen (GET/POST)

- Zugriff auf die Formulardaten über superglobale Arrays ($_GET, $_POST, $_REQUEST)
- Superglobale Arrays sind assoziative Arrays
- Schlüssel ist immer der Inhalt des name-Attributs im HTML

PHP - Grundlagen

Übungen:

1. Erstellen Sie rechts dargestelltes Formular.

2. Geben Sie die übermittelten Daten beim Senden innerhalb desselben Dokuments aus.

PHP - Grundlagen

Datenübergabe per URL:

```
<a href="skript.php?var=val&var2=val2">send data</a>
```

?	Trennung zwischen URL und Daten (Querystring)
&	Trennung der Variablen- / Wertepaare
=	Trennung zwischen Variablenname und Wert

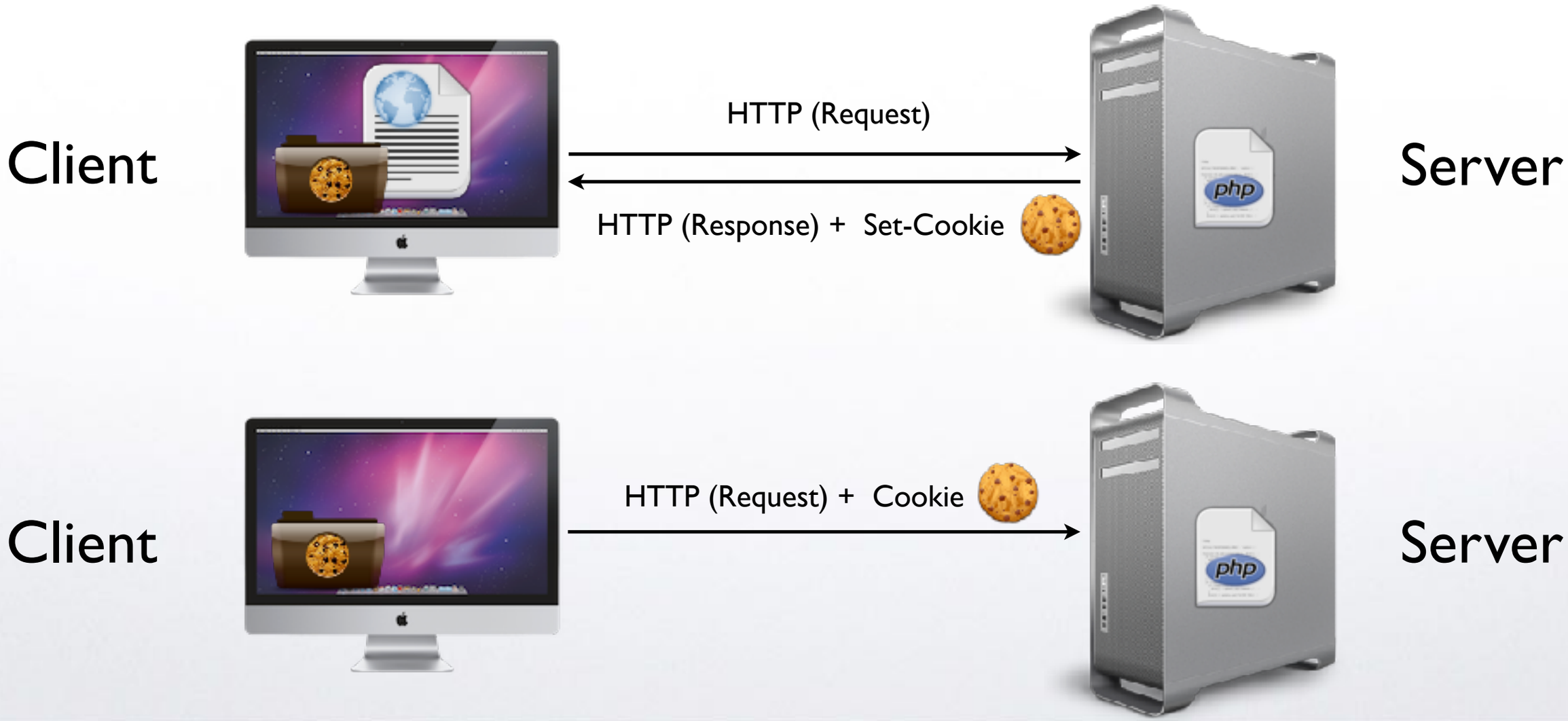
- Zugriff auf Variablen mittels $_GET['var']
- Zugriff auf den kompletten Querystring mittels $_SERVER['QUERY_STRING']
- Nachteile:
  - Störende Escape- und Sonder-Zeichen müssen behandelt werden (URL-Kodierung)
  - Browser akzeptieren maximal 2000 Zeichen (URL + Querystring)
  - Bei gleichlautenden Variablen wird nur der letzte Wert übernommen

PHP - Grundlagen

Cookies - Allgemein:

- Cookies: Mechanismus, den serverseitige Verbindungen nutzen können, um clientseitig Informationen speichern und auslesen zu können
- Erfunden von Netscape 1994
- Sollten die Unzulänglichkeiten von HTTP umgehen
- Vorverurteilung durch Artikel von Jon Udell (BYTE 03/1997)
  - Server kann private Daten vom Computer lesen
  - Andere Server können auf Cookies (damit auch sensible Daten wie Kennwörter) zugreifen
- Richtigstellung der Aussagen (BYTE 05/1997) ist in der Öffentlichkeit untergegangen

PHP - Grundlagen



PHP - Grundlagen

Cookies setzen (HTTP):

```
Set-Cookie: name=[Name];expires=[Datum];path=[Pfad];domain=[Domain];secure
```

name	(Pflichtfeld) Enthält Namen + Inhalt (Verbotene Zeichen: Kommas, Semikolons, Leerzeichen). URL-Kodierung des Inhalts ist sinnvoll.
expires	(Optional) Lebensdauer des Cookies (Cookie wird nach Ablauf gelöscht)
domain	(Optional) Cookie wird nur an den Server gesendet, wenn das Feld passt. Teilmatch reicht. Default: Hostname. Muss mindestens 3 Punkte enthalten.
path	(Optional) Pfad der URL (nach Domain). Cookie wird nur an den Server gesendet, wenn das Feld mit der URL der Anfrage übereinstimmt
secure	(Optional) Cookie wird nur über HTTPS übertragen

PHP - Grundlagen

Cookies in der HTTP-Anfrage:

```
Cookie: name1=[Wert1]; name2=[Wert2]
```

- Alle relevanten Cookies werden vom Browser in der HTTP-Anfrage integriert
- Es werden nur Name und Inhalt der Cookies übertragen
- Mehrere Cookies werden durch Semikolon getrennt
- Server kann ein Cookie löschen, indem der Parameter "expire" auf ein vergangenes Datum gesetzt wird

PHP - Grundlagen

Cookies setzen (PHP):

```
<?php setcookie(string name, [string value], [int expire],
                [string path], [string domain], [bool secure]); ?>
```

- Speichert ein Cookie auf dem Client
  - name: Name des Cookies
  - value: Inhalt des Cookies
  - expire: Lebensdauer des Cookies
  - path: Senden des Cookies mit Anfragen an den angegebenen URL-Pfad
  - domain: Senden des Cookies mit Anfragen die angegebene Domain
  - secure: Senden des Cookies nur bei Verwendung von HTTPS

PHP - Grundlagen

Cookies - Beispiel:

```php
<?php
    setcookie("color", $_POST['set_color'], time() + 180);
    if (isset($_COOKIE['color'])) {
        echo $_COOKIE['color'];
    }
?>
```

- Setzen des Cookies mit setcookie()
- Zugriff auf Cookies durch superglobales Array $_COOKIE

PHP - Grundlagen

Sessionverwaltung - Allgemein:

- Problem: HTTP ist verbindungslos. Server vergisst alle Informationen nach jedem Aufruf.
- Lösungsansätze:
  - Übertragen der Informationen über "hidden fields" (Client)
  - Speichern der Informationen in Cookies (Client)
  - Übertragen der Informationen per URL (GET) (Client)
  - Speichern der Informationen in Dateien (Server)
  - Speichern der Informationen in eine Datenbank (Server)

PHP - Grundlagen

Manuelle Sessionverwaltung:

```
<?php
    // Wurde Session-ID übertragen?
    if (!isset($_POST['session'])) {
        // Neue Session-ID erzeugen
        $id = "ID" . $_SERVER['REMOTE_ADDR'] . time();
        $session = md5($id);
    } else {
        // Session-ID, die über hidden field übertragen wurde, benutzen
        $session = $_POST['session'];
    }
?>
```

- Indentifikation des Benutzers durch eine eindeutige Session-ID
- Speicherung der Session-ID mit Hilfe einer der genannten Methoden

PHP - Grundlagen

Sessionverwaltung - PHP:

- Konfiguration von Sessions in php.ini
- PHP-Sessionverwaltung basiert auf Cookies oder alternativ auf URL-Übergabe (GET)
- Sessionmodul kann auch durch ein eigenes ersetzt werden (damit können alle Methoden verwendet werden)
- Session-Daten werden in temporären Dateien gecached

PHP - Grundlagen

Session-Funktionen:

```
<?php session_start(); ?>
```

- Initialisierung/Wiederaufnahme einer Session (Aufruf bei jedem Skript!)

```
<?php session_destroy(); ?>
```

- Löscht alle Daten, die in Verbindung mit der aktuellen Session stehen

```
<?php session_name([string name]); ?>
```

- Holt oder Setzt den Namen der aktuellen Session (default: "PHPSESSID")

```
<?php session_save_path([string path]); ?>
```

- Holt oder Setzt den aktuellen Speicherort der Session-Daten

PHP - Grundlagen

Session-Funktionen:

```
<?php session_id([string id]); ?>
```

- Holt oder Setzt die aktuelle Session-ID (automatisch erzeugt bei session_start())

```
<?php session_register(mixed name, [mixed value]); ?>
```

- Setzt Variable(n) in aktueller Session

```
<?php session_unregister([string name]); ?>
```

- Löscht Variable in aktueller Session

```
<?php session_is_registered([string name]); ?>
```

- Prüft, ob eine Variable in der aktuellen Session registriert ist

PHP - Grundlagen

Session-Funktionen:

```
<?php session_get_cookie_params(); ?>
```

- Liefert die Parameter des Session-Cookies

```
<?php session_set_cookie_params(int expire, [string domain],
                                [string path], [bool secure]); ?>
```

- Setzt die Parameter des Session-Cookies

```
<?php session_cache_expire([string expire]); ?>
```

- Holt oder Setzt die Cache-Verfallszeit (Default: 180 Min)
- Wird bei jedem Aufruf von session_start() auf Defaultwert gesetzt

PHP - Grundlagen

Sessionverwaltung - PHP:

```php
<?php
    session_start();
    $sessionid = session_id();
    if (isset($_SESSION['counter'])) {
        // Session existiert
        $counter = ((int) $_SESSION['counter']) + 1;
    } else {
        // Session wurde neu initialisiert
        $counter = 1;
    }
    $_SESSION['counter'] = $counter;
?>
```

- Zugriff auf Session-Daten über superglobales Array $_SESSION
- Zuweisung von Daten auch über superglobales Array $_SESSION möglich (alternativ zu session_register())

PHP - Grundlagen

Übungen:

1. Erweitern Sie die Page aus der letzten Übung so, dass erst ein Loginformular erscheint. Die anderen Inhalte sollen erst nach erfolgreichem Login erscheinen. Implementieren Sie das Loginverfahren, indem Sie den Username und das Passwort statisch im Programmcode hinterlegen.

2. Hinterlegen Sie Name, Vorname und Email statisch im Programmcode und speichern Sie diese nach dem Login als Sessionvariablen. Diese Felder sollen im Formular immer vorausgefüllt sein.

PHP - Grundlagen

Dateizugriff - Allgemein:

- Anwendungsfälle:
  - Schreiben von Logdateien
  - Ablage von Formulardaten
  - Speicherung von dynamischen Inhalten (alternative zur Datenbank)
- Ablauf des Dateizugriffs:
  1. Öffnen der Datei
  2. Lesen / Schreiben von Daten
  3. Schließen der Datei

PHP - Grundlagen

File-Funktionen:

```
<?php readfile(string filename); ?>
```

- Liest Dateiinhalt und schreibt ihn in den Ausgabepuffer (impliziert echo)

```
<?php file(string filename, [int flags]); ?>
```

- Liest komplette Datei zeilenweise in ein Array

  Flags (können mit "|" verkettet werden):

  FILE_IGNORE_NEW_LINES

  Zeilenende (\r\n) nicht am Ende jedes Array-Elements einfügen

  FILE_SKIP_EMPTY_LINES

  Leere Zeilen überspringen

PHP - Grundlagen

File-Funktionen:

```
<?php fopen(string filename, string mode); ?>
```

- Öffnet eine Datei und gibt ein Handle zurück

  Modi:

r	Lesen (Zeiger am Dateianfang)	w+	Wie "w" + Lesen
r+	Wie "r" + Schreiben	a	Wie "w" (Zeiger am Dateiende)
w	Schreiben (Erzeugt/Löscht Datei)	a+	Wie "a" + Lesen
b	Kombinierbar mit allen anderen Modi (Für Binärdateien)		

```
<?php fclose(resource handle); ?>
```

- Schließt Datei und gibt sie anderen Prozessen frei

PHP - Grundlagen

File-Funktionen:

```
<?php fgets(resource handle, [int length]); ?>
```

- Liest Dateiinhalt bis angegebene Länge, Zeilenende oder Dateiende

```
<?php fread(resource handle, [int length]); ?>
```

- Liest Dateiinhalt bis angegebene Länge oder Dateiende

```
<?php fputs(resource handle, string s,[int length]); ?>
```

- Schreibt Daten in Datei (optional bis definierter Länge)

```
<?php fwrite(resource handle, string s,[int length]); ?>
```

- Alias für fputs(), daher gleiches Verhalten

PHP - Grundlagen

Dateizugriffe - Beispiele:

```php
<?php
    $file = "path/file.txt";
    $fhandle = fopen($file, "r");
    while ($line = fgets($fhandle))
    {
        echo $line . "<br>";
    }
    fclose($fhandle);
?>
```

Aus Datei lesen

```php
<?php
    $logfile = "logdata/file.log";
    $fhandle = fopen($logfile, "a");
    $d = sprintf("%s, %s\n",
              date("d.M.Y h:m:s),
              $_SERVER['REMOTE_ADDR']);
    fputs($fhandle, $d);
    fclose($fhandle);
?>
```

In Datei schreiben

PHP - Grundlagen

File-Funktionen:

```
<?php is_[file|dir|link](string filename); ?>
```

- Prüft ob Pfad eine Datei, Verzeichnis, oder Verknüpfung (bei Win: Datei) ist

```
<?php filemtime(string filename); ?>
```

- Gibt Datum der letzten Änderung in Unixzeit zurück

```
<?php fileatime(string filename); ?>
```

- Gibt Datum des letzten Zugriffs in Unixzeit zurück

PHP - Grundlagen

File-Funktionen:

```
<?php filetype(string filename); ?>
```

- Liefert den Typ der übergebenen Datei (fifo, char, block, link, dir, file)

```
<?php filesize(string filename); ?>
```

- Liefert die Größe der übergebenen Datei in Byte

```
<?php stat(string filename); ?>
```

- Liefert Array mit 13 Dateieigenschaften/Statistiken
- (Windows unterscheidet sich von Unix)

PHP - Grundlagen

Beispiel - Dateigröße:

```
<?php
    function get_real_volume($v = 0) {
        if ($v > pow(2, 10)) {
            if ($v > pow(2, 30)) {
                $r = (integer) ($v / pow(2, 30));
                $r .= " GB";
            } elseif ($v > pow(2, 20)) {
                $r = (integer) ($v / pow(2, 20));
                $r .= " MB";
            } else {
                $r = (integer) ($v / pow(2, 10));
                $r .= " KB";
            }
        } else {
            $r = (string) $v . " Byte";
        }
        return $r;
    }
    echo get_real_volume(filesize("path/file"));
?>
```

PHP - Grundlagen

Dateioperationen:

```
<?php copy(string source, string destination); ?>
```

- Kopiert die angegebene Datei von source nach destination

```
<?php rename(string old, string new); ?>
```

- Bennent die angegebene Datei von old nach new um (auch zum Veschieben)

```
<?php unlink(string filename); ?>
```

- Löscht die angegebene Datei

PHP - Grundlagen

Verzeichnisse:

```
<?php mkdir(string dirname, [int mode], [bool recursive]); ?>
```

- Erstellt ein/mehrere (rekursiv) Verzeichnisse optional mit bestimmten Rechten

```
<?php opendir(string dirname); ?>
```

- Öffnet ein Verzeichnis und gibt ein Handle zurück

```
<?php closedir(ressource handle); ?>
```

- Schließt ein Verzeichnis

```
<?php rmdir(string dirname); ?>
```

- Löscht das angegebene Verzeichnis

PHP - Grundlagen

Übungen:

1. Hinterlegen Sie Username, Passwort, Vorname, Nachname und Email in eine Datei getrennt durch den String "@@@".

2. Erstellen Sie mehrere User.

3. Erweitern Sie das Login so, dass es die Daten aus der Datei bezieht, sodass die richtigen Daten im Formularfeld vorausgefüllt werden.

PHP - Grundlagen

Fileupload - Formular:

```
<form enctype="multipart/form-data" action="upload.php" method="post">
    <input type="hidden" name="max_size" value="1000">
    Ihre Dateiauswahl: <input name="upfile" type="file">
    <input type="submit" value="Datei senden">
</form>
```

- Funktioniert über HTTP POST (Erweiterung in RFC 1867)
- Zusätzliche Informationen können z.B. über "hidden fields" übertragen werden
- Auswahlfenster für die Dateiauswahl wird vom Browser gehändelt

PHP - Grundlagen

Fileupload - Zugriff:

```
<?php
    echo "Name (tempraer):" . $upfile      . "<br>";
    echo "Name (original):" . $upfile_name . "<br>";
    echo "Groesse:"         . $upfile_size . "<br>";
    echo "Typ:"             . $upfile_type . "<br>";

    echo "Name (tempraer):" . $_FILES['upfile']['tmp_name'] . "<br>";
    echo "Name (original):" . $_FILES['upfile']['name']     . "<br>";
    echo "Groesse:"         . $_FILES['upfile']['size']     . "<br>";
    echo "Typ:"             . $_FILES['upfile']['type']     . "<br>";
?>
```

- Attribut "name" des Inputfeldes definiert die Variablennamen in PHP
- Universeller Zugriff auf alle hochgeladenen Dateien mit dem superglobalen Array $_FILES

PHP - Grundlagen

Fileupload - Speichern:

```
<?php
    foreach ($_FILES as $fieldname => $arrfile) {
        if (!($arrfile['size'] > $_POST['max_size'])) {
            $tmpfile  = $arrfile['tmp_name'];
            $filename = $arrfile['name'];
            move_uploaded_file($tmpfile, "ulpath/$filename");
        }
    }
?>
```

- Datei wird erstmal in ein temporäres Verzeichnis geladen (definiert in php.ini)
- Dateien im temporären Verzeichnis werden am Ende des Skripts gelöscht. Die Dateien müssen nach dem Upload in das Zielverzeichnis verschoben werden mit copy() oder move_uploaded_file() (besser)

PHP - Grundlagen

Übungen:

1. Erweitern Sie das Formular um ein File-Upload-Feld.

2. Verschieben Sie die hochgeladene Datei in den Ordner "uploads".

PHP - Grundlagen

Mail-Funktion:

```
<?php mail(string to, string subject, string msg, [string
addheaders], [string $add_params]); ?>
```

- Verschickt eine Email
- Pflichtfelder: Empfänger, Betreff, Inhalt
- Optional: Definition weiterer Header des SMTP-Protokolls (z.B. Absender)
- Gibt einen boolschen Wert zurück, der aussagt, ob das Versenden der Email funktioniert hat

PHP - Grundlagen

Email verschicken:

```
<?php
    $rcv      = $_POST['rcv'];
    $subject  = "Kontaktanfrage!";
    $from     = $_POST['email'];
    $name     = $_POST['name'];
    $surname  = $_POST['surname'];
    $msg      = $_POST['msg'];

    $content  = "
Eine automatisch generierte Email von $name $surname ($rcv).
Message war:
$msg
";

    mail($from, $subject, $content, "from: info@domain.de\r\nreply-to: ".$rcv);
?>
```

PHP - Grundlagen

HTML-Email verschicken:

```
<?php
    $message = '
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Newsletter</title>
</head>
<body><p>Ein kleiner HTML Newsletter.</p></body>
</html>';

    $arr_rcv = array("user1@mail.de", "user2@mail.com", "user3@mail.org");
    $subject = "Unser erster HTML-Newsletter";
    $xtra = "MIME-Version: 1.0\r\n";
    $xtra .= "Content-Type: text/html; charset=iso-8859-1\r\n";
    $xtra .= "From: Ich <ich@me.com>\r\n";
    $xtra .= "Bcc: $bcc\r\n";

    foreach ($arr_rcv as $rcv) mail($rcv, $subject, $message, $xtra);
?>
```

PHP - Grundlagen

Übungen:

1. Erweitern Sie das Formular um eine Checkbox "als Email versenden".

2. Schicken Sie bei selektierter Checkbox eine Email, die alle im Formular angegebenen Informationen enthält.

PHP - Grundlagen

Regexp-Funktionen:

```
<?php preg_match(string p, string s); ?>
```

- Sucht das erste Vorkommen von p in s.

```
<?php preg_match_all(string p, string s); ?>
```

- Sucht alle Vorkommen von p in s.

```
<?php preg_replace(mixed p, mixed r, mixed s, [integer m]); ?>
```

- Ersetzt alle Vorkommen von p in s durch r (maximal m mal).

PHP - OO

Klassendefinition:

```
<?php
   class calc {
      var $factor;

      function into($value) {
         return $value / $this->factor;
      }

      function from($value) {
         return $value * $this->factor;
      }
   }

   $c = new calc;
   $c->factor = 1.95583;
   echo "100 DM sind ".$c->into(100)." Euro.";
?>
```

PHP - OO

Attribute und Methoden:

```
<?php
   class calc {
      var $factor;

      function into($value) {
         return $value / $this->factor;
      }

      function from($value) {
         return $value * $this->factor;
      }
   }

   $c = new calc;
   $c->factor = 1.95583;
   echo "100 DM sind ".$c->into(100)." Euro.";
?>
```

PHP - OO

Instanziierung:

```
<?php
   class calc {
      var $factor;

      function into($value) {
         return $value / $this->factor;
      }

      function from($value) {
         return $value * $this->factor;
      }
   }

   $c = new calc;
   $c->factor = 1.95583;
   echo "100 DM sind ".$c->into(100)." Euro.";
?>
```

PHP - OO

Attributzugriff und Methodenaufruf:

```
<?php
   class calc {
      var $factor;

      function into($value) {
         return $value / $this->factor;
      }

      function from($value) {
         return $value * $this->factor;
      }
   }

   $c = new calc;
   $c->factor = 1.95583;
   echo "100 DM sind ".$c->into(100)." Euro.";
?>
```

PHP - OO

Zugriff innerhalb einer Instanz:

```
<?php
   class calc {
      var $factor;

      function into($value) {
         return $value / $this->factor;
      }

      function from($value) {
         return $value * $this->factor;
      }
   }

   $c = new calc;
   $c->factor = 1.95583;
   echo "100 DM sind ".$c->into(100)." Euro.";
?>
```

PHP - OO

Konstruktoren:

```
<?php
   class calc {
      var $factor;

      function __construct($init) {
         $this->factor = $init;
      }

      function into($value) {
         return $value / $this->factor;
      }

      function from($value) {
         return $value * $this->factor;
      }
   }

   $c = new calc(1.95583);
   echo "100 DM sind ".$c->into(100)." Euro.";
?>
```

PHP - OO

Destruktoren:

```
<?php
   class calc {
      var $factor;

      function into($value) {
         return $value / $this->factor;
      }

      function from($value) {
         return $value * $this->factor;
      }

      function __destruct($init) {
         // Aktionen beim Löschen des Objekts
         …
      }
   }
?>
```

PHP - OO

Sichtbarkeit:

public [, var]	Zugriff von überall aus möglich
protected	Zugriff von der eigenen und den erbenden Klassen (nicht vom Objekt aus)
private	Zugriff nur innerhalb der Klasse (nicht vom Objekt aus)

```
<?php
   class example {
      private $text;

      protected function print($value) {
         …
      }
   }
?>
```

PHP - OO

Statische Methoden/Attribute:

```
<?php
   class calc {
      static factor = 1.95583;

      static function into($value) {
         return $value / self::factor;
      }

      static function from($value) {
         return $value * self::factor;
      }
   }

   echo "100 DM sind ".calc::into(100)." Euro.";
   echo "Umrechnungskurs ist ".calc::factor.".";
?>
```

PHP - OO

Vererbung:

```
<?php
   class calc {
      protected factor;
      function __construct($init) {
         $this->factor = $init;
      }

      function convert($value) {
         return $value / $this->factor;
      }
   }
   class dm_calc extends calc {
      function __construct() {
         parent::__construct(1.95583);
      }
   }
?>
```

PHP - OO

Interfaces:

```
<?php
   interface calc {
      public function into($value);
      public function from($value);
   }

   class dm_calc implements calc {
     private $factor = 1.95583;
     function into($value) {
        return $value / $this->factor;
     }

     function from($value) {
        return $value * $this->factor;
     }
   }
?>
```

PHP - OO

Abstrakte Klassen:

```
<?php
   abstract class calc {
      protected $factor;
      abstract public function into($value);
      abstract public function from($value);
      public function get_factor() {…}
   }

   class dm_calc extends calc {
     protected $factor = 1.95583;
     function into($value) {
        return $value / $this->factor;
     }

     function from($value) {
        return $value * $this->factor;
     }
   }
?>
```

PHP - OO

Finale Klassen:

```
<?php
   final class calc {
      var factor;

      function into($value) {
         return $value / $this->factor;
      }

      function from($value) {
         return $value * $this->factor;
      }
   }
?>
```

- Unterdrückung der Vererbung

PHP - OO

De-/Serialisierung:

```
<?php serialize(object $o); ?>
```

- Serialisierung eines Objektes in einen String.

```
<?php unserialize(string $s); ?>
```

- Deserialisierung eines Objektes von einem String in den ursprünglichen Zustand.

```
<?php __sleep(); ?>
```

- Methode des zu serlialisierenden Objekts zum Steuern der Sterilisierung.

```
<?php __wakeup(); ?>
```

- Quasi Konstruktor bei der Deserialisierung

PHP - OO

Hilfsfunktionen:

```
<?php method_exists(object $o, string m); ?>
```

- Prüft, ob Methode m in Objekt o vorhanden ist.

```
<?php is_subclass_of(object $o, string $c); ?>
```

- Prüft, ob Objekt o eine Unterklasse der Klasse c ist.

```
<?php get_class_methods(string $c); ?>
```

- Gibt alle Methoden der Klasse c zurück.

```
<?php is_a(string $c); ?>
```

- Prüft, ob Objekt o eine Instanz der Klasse c ist.

PHP - OO

Hilfsfunktionen:

```
<?php get_class(object $o); ?>
```

- Ergibt den Namen der Klasse des Objekts o.

```
<?php get_parent_class(object $o); ?>
```

- Ergibt den Namen der übergeordneten Klasse des Objekts o.

```
<?php get_class_vars(string $c); ?>
```

- Array mit Namen aller Attribute der Klasse c.

```
<?php get_object_vars(object $o); ?>
```

- Array mit Namen aller tatsächlich genutzten Attribute des Objekts o.

PHP - OO

besondere Methoden:

```
<?php __clone(); ?>
```

- Wird beim Kopieren eines Objekts aufgerufen.

```
<?php __toString(); ?>
```

- Beschreibung des Objekts als String.

```
<?php __get(string $p); ?>
```

- Universelle Get-Methode für ein Attribut p.

```
<?php __set(string $p, mixed $v); ?>
```

- Universelle Set-Methode für ein Attribut p. Setzt den Wert von p auf v.

PHP - DB

Verbindung zu MySQL:

```
<?php mysqli_connect(string $h, string $u, string $p, string $d); ?>
```

- MySQL-Verbindung zu Host h (DB d) mit User u und Passwort p. Liefert Handler.

```
<?php mysqli_set_charset(mysqli $h, string $c); ?>
```

- Setzt das Encoding für die Kommunikation über die Verbindung h auf c.

```
<?php mysqli_connect_errno(); ?>
```

- Liefert den Fehlercode des letzten Verbindungsversuchs (mysqli_connect).

```
<?php mysqli_connect_error(); ?>
```

- Liefert die Fehlerbeschreibung des letzten Verbindungsversuchs (mysqli_connect).

PHP - DB

Verbindung zu MySQL:

```
<?php mysqli_query(mysqli $h, string $sql); ?>
```

- Setzt eine SQL-Anfrage s über die Verbindung zu Host h ab. Liefert einen Handler.

```
<?php mysqli_close(mysqli $h); ?>
```

- Schließt die MySQL-Verbindung h.

```
<?php mysqli_errno(mysqli $h); ?>
```

- Liefert den Fehlercode der letzten Abfrage über Verbindung h.

```
<?php mysqli_error(mysqli $h); ?>
```

- Liefert die Fehlerbeschreibung der letzten Abfrage über Verbindung h.

PHP - DB

Ergebnisse von MySQL-Abfragen:

```
<?php mysqli_fetch_field(mysqli_result $h); ?>
```

- Liefert ein Objekt mit der Definition der aktuellen Spalte im Ergebnis.

```
<?php
  $mysqli = connect_mysql();

  $res = mysqli_query($mysqli, "SELECT *
FROM test") or die("Error: " . $mysqli-
>error);

  $arr = Array();
  while ($r=mysqli_fetch_field($res)) {
    array_push($arr, $r);
  }

  mysqli_close($mysqli);
?>
```

```
stdClass Object
(
    [name] => id
    [orgname] => id
    [table] => test
    [orgtable] => test
    [def] =>
    [db] => test
    [catalog] => def
    [max_length] => 1
    [length] => 11
    [charsetnr] => 63
    [flags] => 49667
    [type] => 3
    [decimals] => 0
)
```

PHP - DB

Ergebnisse von MySQL-Abfragen:

```
<?php mysqli_fetch_assoc(mysqli_result $h); ?>
```

- Liefert ein assoziatives Array der Zeile oder NULL, falls keine Zeilen übrig sind.

```
<?php
  $mysqli = connect_mysql();

  $res = mysqli_query($mysqli, "SELECT *
FROM test") or die("Error: " . $mysqli-
>error);

  $arr = Array();
  while ($r=mysqli_fetch_assoc($res)) {
    array_push($arr, $r);
  }

  mysqli_close($mysqli);
?>
```

```
Array
(
    [id] => 1
    [name] => Di Caprio
    [vorname] => Leonardo
    [email] => leodicap@gmail.com
)
```

PHP - DB

Ergebnisse von MySQL-Abfragen:

```
<?php mysqli_fetch_array(mysqli_result $h); ?>
```

- Liefert ein Array (indiziert und assoziativ) der Zeile oder NULL, falls keine Zeilen übrig sind.

```
<?php
  $mysqli = connect_mysql();

  $res = mysqli_query($mysqli, "SELECT *
FROM test") or die("Error: " . $mysqli-
>error);

  $arr = Array();
  while ($r=mysqli_fetch_array($res)) {
    array_push($arr, $r);
  }

  mysqli_close($mysqli);
?>
```

```
Array
(
    [0] => 1
    [id] => 1
    [1] => Di Caprio
    [name] => Di Caprio
    [2] => Leonardo
    [vorname] => Leonardo
    [3] => leodicap@gmail.com
    [email] => leodicap@gmail.com
)
```

PHP - DB

Ergebnisse von MySQL-Abfragen:

```
<?php mysqli_fetch_row(mysqli_result $h); ?>
```

- Liefert ein Array der Zeile oder NULL, falls keine Zeilen übrig sind.

```
<?php
  $mysqli = connect_mysql();

  $res = mysqli_query($mysqli, "SELECT *
FROM test") or die("Error: " . $mysqli-
>error);

  $arr = Array();
  while ($r=mysqli_fetch_row($res)) {
    array_push($arr, $r);
  }

  mysqli_close($mysqli);
?>
```

```
Array
(
    [0] => 1
    [1] => Di Caprio
    [2] => Leonardo
    [3] => leodicap@gmail.com
)
```

PHP - DB

MySQL Prepared Statements:

```
<?php mysqli_prepare(string $h, string $s); ?>
```

- Bereitet eine SQL-Anfrage s über die Verbindung zu Host h vor. Liefert einen Handler.

```
<?php mysqli_stmt_bind_param(mysqli_stmt $s, string $t, mixed v); ?>
```

- Bindet die Parameter v vom Typ t an das Statement s.

```
<?php mysqli_stmt_execute(mysqli_stmt $s); ?>
```

- Führt das Statement s aus. Liefert einen Handler mit dem Ergebnis.

PHP - DB

MySQL Prepared Statements:

```
<?php

$mysqli = connect_mysql();
if (!($stmt = mysqli_prepare($mysqli, "INSERT INTO test(name, vorname)
VALUES (?, ?)"))) {
    echo "Prepare failed: (" . $mysqli->errno . ") " . $mysqli->error;
}

if (!mysqli_stmt_bind_param($stmt, "ss", $name , $vorname)) {
    echo "Binding parameters failed: (" . mysqli_stmt_errno($stmt) . ")
" . mysqli_stmt_error($stmt);
}

if (!($res = mysqli_stmt_execute($stmt))) {
    echo "Execute failed: (" . mysqli_stmt_errno($stmt) . ") " .
mysqli_stmt_error($stmt);
}

mysqli_close($mysqli);

?>
```

PHP - DB

MySQL Stored Procedures:

```
DROP PROCEDURE IF EXISTS <p>;
```

- Löscht eine existierende Prozedur auf dem MySQL-Server mit Namen p.

```
CREATE PROCEDURE <p>(IN <v> <t>) BEGIN <q>; END;
```

- Erzeugt eine Prozedur p auf dem Server Mit dem Parameter v vom Typ t, welche die SQL-Abfrage q ausführt.

```
CALL <p>(<v>);
```

- Ruft die Prozedur p mit dem Parameter v auf und liefert das Ergebnis.

PHP - DB

MySQL Stored Procedures:

```
<?php

$mysqli = connect_mysql();
if (
   !mysqli_query($mysqli, "DROP PROCEDURE IF EXISTS p") ||
   !mysqli_query($mysqli, "CREATE PROCEDURE p(IN id_val INT) BEGIN SELECT
* FROM test WHERE id=id_val; END;“)
) {
    echo "Stored procedure creation failed: (" . mysqli_errno($mysqli) .
") " . mysqli_error($mysqli);
}

if (!($res = mysqli_query($mysqli, "CALL p(".$id.")"))) {
    echo "CALL failed: (" . mysqli_errno($mysqli) . ") " .
mysqli_error($mysqli);
}

$mysqli_close($mysqli);

?>
```

PHP - OO/DB

Übungen:

1. Erstellen Sie eine Test-Datenbank und eine Test-Tabelle.

2. Erstellen Sie eine PHP Seite, welche die DB-Tabelle tabellarisch anzeigt. Verwenden Sie dafür die objektorientierte Variante von mysqli (siehe php.net).

3. Erlauben Sie das Filtern der Tabelle anhand einer Spalte. Benutzen Sie dafür sowohl „prepared statements“ als auch „stored procedures“.