

Am Anfang war die "normale" Webseite ...

- besteht aus reinem HTML
- ist statisch, muß handgepflegt werden
- kein dynamisches Anpassen an Benutzereingaben oder andere sich ändernde Inhalte (Seitencounter, Datenbanken)

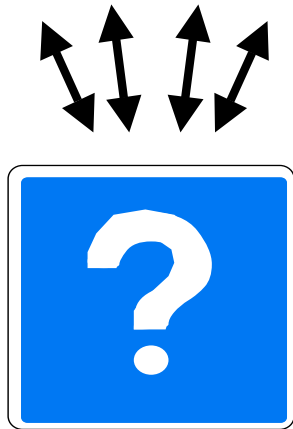


... dann kam das CGI-Programm

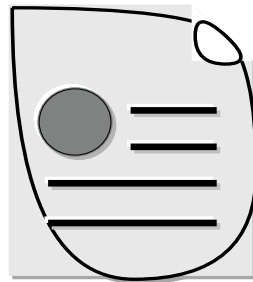
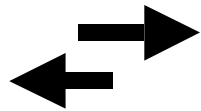
- nimmt Daten an (Suchworte), tut etwas damit (Datenbankabfrage) und
- gibt eine fertig formatierte Webseite aus
- abgekoppelt von der Webseite
- Änderungen im Weblayout sind am Programmcode vorzunehmen
- läuft auf dem Webserver als extra Programm/Prozess

Webseite und CGI-Programm

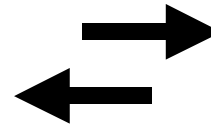
Zugriff auf Datenquellen



CGI-Programm



Webseite

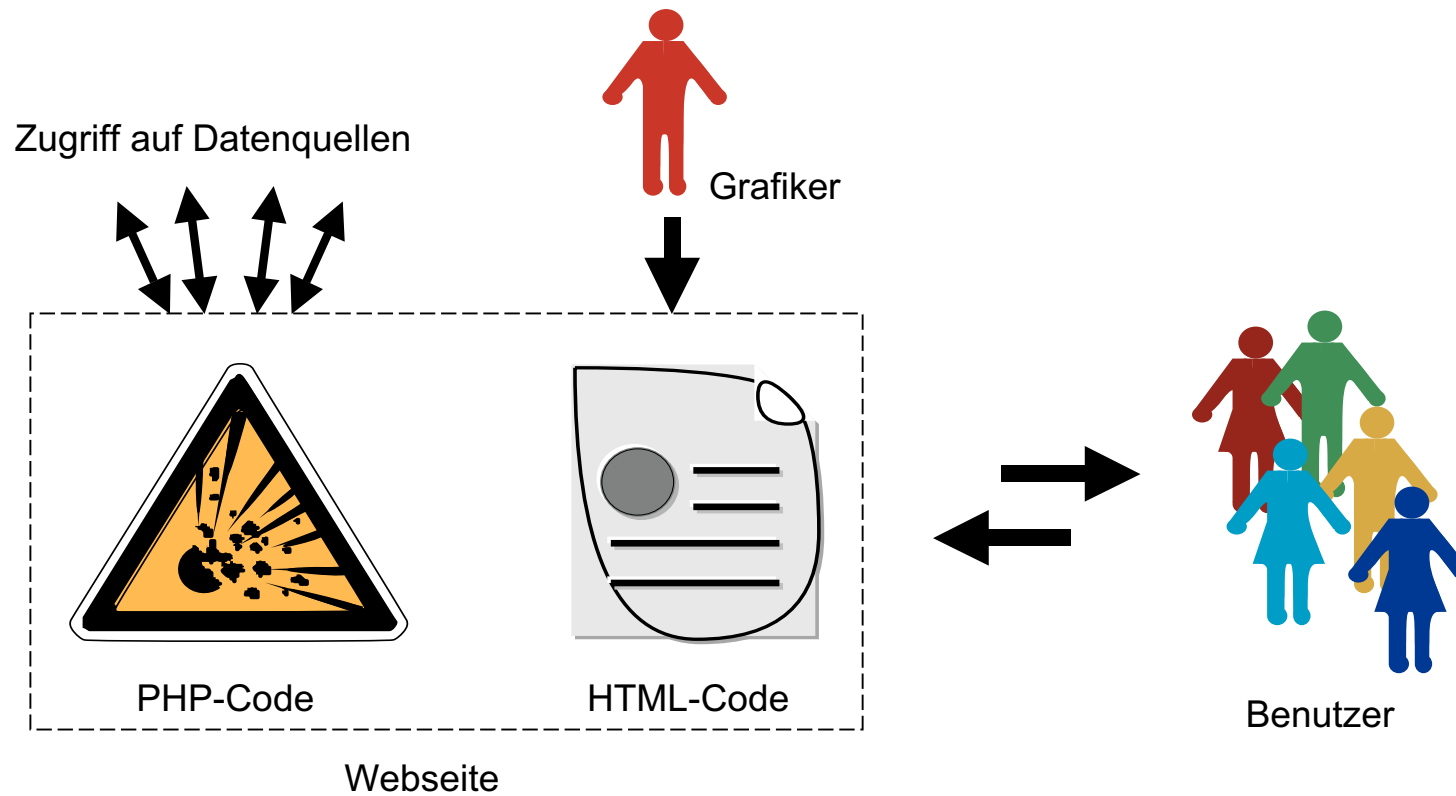


Benutzer

... jetzt gibt es PHP

- Programmfunktionalität ist in die Webseite eingebettet
- Zugehörigkeit von Programmcode zur Webseite ist eindeutig
- Aber: bei großen Programmen schnell unübersichtlich
Lösung:
- Templates: Aufteilung von HTML (Layout) und Programmcode (Funktionalität)

Webseiten und PHP



Was PHP kann

- Zwischen Webserver und Datenbanken vermitteln
- umfangreiche Datenmanipulationen
- Dateizugriffe
- Internet-Protokolle (FTP, HTTP, LDAP, ...)
- Grafiken erstellen (GIF, JPEG, PNG)
- PDF-Dateien erstellen
- Dateien auf den Server übertragen (file upload)

generell: Webseiten dynamisch gestalten

Was PHP nicht kann

- Keine ständig laufenden Hintergrundprozesse, die z.B. Client-Server-Verbindungen bedienen oder auf Ereignisse reagieren (Application Server)

Wozu kann PHP genutzt werden

- Online-Shop, Terminplaner, CMS, weitere Datenbank-basierte Anwendungen
- PHP kann nicht für Aktionen auf der Seite des Clienten genutzt werden (wie z.B. JavaScript)

generell: jede Server-basierte Webapplikation

Syntax in PHP

- `<?php // Das ist mit XML kompatibel ?>`
- `<? /* Sowas mag SGML */ ?>`
- `<% // Sieht aus wie ASP von Microsoft %>`
- `<script language="php">`
 `// Frontpage freut sich darüber`
 `</script>`
- `echo "Wo ist das Ende ?";`



Datentypen in PHP

- Integer, Real, String
- Konstanten
- Arrays (skalar, assoziativ)
- Objekte

Datentypen : Variablen

- `$integer_var = 1;`
- `$real_var = 1.234;`
- `$string_var = "Text ist \"fast\" kurz";`
- `$html_string = '';`
- `$msg = 'Hallo ' . $vorname . $nachname;`

Text mit doppelten Anführungszeichen sollte zwischen einfache Anführungszeichen gestellt werden und umgedreht.

Alternativ können solche Zeichen mit Backslash angeführt werden.

Texte und Inhalte von Variablen werden mit dem Punkt verbunden.



Datentypen : Konstanten

- `define("CONSTANTIN", "Hello world.");`
- Vordefiniert sind z.B. `TRUE` und `FALSE`
- Mehr in der PHP-Doku

Datentypen: Arrays

- `$a[0] = "abc";`
- `$a[1] = "def";`
- `$formular["name"] = 13;`
- `$formular = array`
 `("name" => "Bernhard",`
 `"vorname" => "Fürst",`
 `"Ort" => "Freiberg")`



Kontrollstrukturen

- `if ... elseif ... else`
- `switch ... case ... default`
- `for`
- `foreach`
- `while`
- `do ... while`
- `break`
- `continue`

Kontrollstrukturen: if

```
if ($a > $b) {  
    print "a is bigger than b";  
} elseif ($a == $b) {  
    print "a is equal to b";  
} else {  
    print "a is smaller than b";  
}
```



Kontrollstrukturen: switch

```
if ($i == 0) {  
    print "i equals 0";  
}  
if ($i == 1) {  
    print "i equals 1";  
}  
if ($i == 2) {  
    print "i equals 2";  
}
```

```
switch ($i) {  
    case 0:  
        print "i equals 0";  
        break;  
    case 1:  
        print "i equals 1";  
        break;  
    case 2:  
        print "i equals 2";  
        break;  
    default:  
        print "Voreinstellung";  
        break;  
}
```

Links die Abfrage mit `if`, rechts alternativ mit `switch`



Kontrollstrukturen: for

```
for ($i = 1; $i <= 10; $i++)  
{  
    print $i;  
}
```



Kontrollstrukturen: foreach

```
foreach($arr as $key => $value)
{
    echo "Key: $key   Value: $value<br>";
}
```

Mit foreach kann sehr komfortabel der Inhalt eines Arrays gelesen werden.



Kontrollstrukturen: while

```
$i = 1;
while ($i <= 10)
{
    print $i;
    $i++;
}
```

Kontrollstrukturen: do while

```
$i = 0;  
do  
{  
    print $i;  
    $i++;  
} while ($i>0);
```



Kontrollstrukturen: `break`

- `break;`
verläßt das aktuelle `for`, `while` oder `switch`
Konstrukt
- `break n;`
verläßt `n` verschachtelte `for`, `while` oder
`switch` Konstrukte

Kontrollstrukturen: `continue`

- `continue;`

In Schleifen: ignoriert folgende Befehle
und setzt am Schleifeniterationenpunkt fort

- `continue n;`

setzt am Schleifeniterationenpunkt der n-ten
äußeren Schleife fort

Operatoren

- Arithmetisch
- Zuweisungen
- Vergleiche
- Bitweises Ein-/Ausschalten
- Inkrement/Dekrement
- Logische Operatoren
- Zeichenkettenop.
- Fehlerkontrolle

Detailliertere Infos in der PHP-Doku.

Funktionen

```
function lehrgang( $input )  
{  
    global $do_nothing;  
  
    $what = $do_nothing + $input;  
    return $what;  
}
```



Datentypen: Objekte definieren

```
class datenbank
{
    var $user = "chef";
    var $pass = "susi";

    function verbinden( $server )
    {
        sql_connect( $server, $this->user, $this->pass );
    }
}
```

Datentypen: Objekte nutzen

```
$sammlung = new datenbank;  
$sammlung->verbinden( "www.tu-freiberg.de" );
```

```
class mitarbeiter_db extends datenbank  
{  
    $user = "kein_chef";  
    $pass = "carola";  
}
```

```
$sammlung = mitarbeiter_db;  
$sammlung->verbinden( "www.tu-freiberg.de" );
```

Klassen sind Baupläne für ein Paket von Variablen und Funktionen. Gebaut werden Objekte, die bequemen und in sich konsistenten Zugriff auf Daten erlauben.



Datentypen: Objekt-Konstruktor

```
class datenbank
{
    var $user = "chef";
    var $pass = "susi";
    var $server = "www.tu-freiberg.de";

    function datenbank( $server )
    {
        sql_connect( $this->server, $this->user, $this->pass );
    }
}
```

Der Konstruktor ist eine Funktion, die den Namen der Klasse trägt. Beim Erzeugen eines Objektes mit `new` wird diese Funktion einmal ausgeführt.

Kommentare

```
/*  
 * Viele Zeilen Kommentar *  
 * blah blah *  
 * Viele Zeilen *  
***/
```

```
// Eine Zeile Kommentar
```

Infos im Netz

- Homepage: <http://www.php.net>
- Manual: <http://www.geo.tu-freiberg.de/docs/php/>
- ausführliche FAQ: <http://www.koehntopp.de/php/>
- PHP und Objekte:
<http://www.koehntopp.de/kris/artikel/data-driven2/>

Lesen in einer MySQL-Datenbank

- Aufgaben des PHP-Scriptes:
 - Verbindungsaufbau zum Datenbankserver mit `mysql_connect()`
 - Auswahl der Datenbank: SQL-Befehl mit `mysql_query()` ausführen.
 - Datenabfrage: SQL-Befehl mit `mysql_query()`
 - Zeilenweises Auslesen der Daten mit einem der `mysql_fetch_*` Befehle und Ausgabe auf der Webseite mit `echo`



Beschreiben einer MySQL-Datenbank

- Aufgaben des PHP-Scriptes:
 - Verbindungsaufbau zum Datenbankserver mit `mysql_connect()`
 - Auswahl der Datenbank: SQL-Befehl mit `mysql_query()` ausführen.
 - Daten schreiben: SQL-Befehl `insert` mit `mysql_query()` ausführen.



Hauptroutine für PHP-Skripte

```
switch( $task )
{
    case „eingabe“:
        eingabe( $name, $email, $telefon);
        break;
    case „ausgabe“:
        ausgabe();
        break;
    default:
        menue_anzeigen();
}
```

Dem Hauptskript werden via \$task Aufgaben benannt, zu denen es in der switch-Konstruktion die entsprechenden Aktionen in Gang setzt.

Übung

- Zusammenlegen der Funktionalität aus `php_11.php` und `php_12.php`
- Leistungsumfang:
 - `function()` verwenden
 - Zentraler Dispatcher mit `switch`
 - HTML-Formular zur Eingabe der Daten
 - HTML-Tabelle zur Ausgabe

Sicherheit für Benutzer und Betreiber

- Herkunft der von außen kommenden Variablen kontrollieren
 - Möglichkeiten: GET, POST, COOKIES, SERVER, SESSION, ENV, FILES, REQUESTS
- Lösungen:
 - Zugriff über globale Variablen `HTTP_*_VARS[]`
 - Ab PHP 4.1: `_*[]` (auch innerhalb von Funktionen ohne „global“ verfügbar)

(* = eine der obigen Möglichkeiten)



Sicherheit für Benutzer und Betreiber

- Richtigkeit der Daten überprüfen
 - Grundsatz: alle Daten von außen (Webbrowser) sind potentiell gefährlich
- Lösung:
 - Im HTML-Formular: `maxlength` nutzen
 - In PHP: Validieren mit Hilfe von Regulären Ausdrücken, bevor die Daten weiter im Programm verwendet werden.



Sessions

- Problem: Variableninhalte und andere interne Programmdateien gehen nach Ablauf der PHP-programmierten Webseite verloren.
- Lösungen:
 - Daten über URL/Formular/Cookie von Seite zu Seite über den Webbrowser weiterreichen: Aufwendig und fehleranfällig, da bei Änderungen der internen Datenstruktur jede URL/Formular geändert werden muß. Dazu müssen Daten bei jedem Neulauf des Programmes erneut validiert werden.
 - PHP-Sessions: Daten werden automatisch in der Session auf dem Server gespeichert und stehen beim nächsten Programmlauf wieder zur Verfügung. Identifizierung der Session durch eine Session ID, die via URL, Formular oder Cookie vom Browser an den Server gesendet wird (automatisch durch PHP).



Datenbankzugriff via Objekt

- Einsatz des Datenbankwrappers Db_SQL aus der PHPLIB.
(<http://phplib.sourceforge.net/>)
- Vorteile:
 - Vereinfachter und vom verwendeten DBMS unabhängiger Zugriff auf Datenbanken.
 - Fehlerbehandlung übernimmt die Db_SQL.
- Alternative: PEAR DB
(<http://pear.php.net/>)



Editieren von Daten einer Datenbank

- Merkmale
 - Jede Funktion des Programms ist in einer eigenen Datei untergebracht (modularer Aufbau)
 - Anzeige in Tabellenform
 - Vorhandene Datensätze sollen editierbar sein
 - Zugriff auf die Datenbank via Objekt



Editieren von Daten einer Datenbank

- Eckpunkte des Programmes:
 - Erweitern der Klasse `Db_SQL` mit den aktuellen Verbindungsdaten
 - `switch`-Konstrukt wertet `$task` aus, und schaltet auf die entsprechende Funktion.
 - In Einzeldateien abgelegte Funktionen für:
 - Ausgabe in HTML-Tabelle inkl. Link zum Editieren pro Datensatz
 - Eingabe von Daten via HTML-Formular
 - Editieren von Daten via HTML-Formular



HTML-Ausgabe via Template

- Problem: in umfangreichen Projekten ist die Vermischung von HTML- und PHP-Code unübersichtlich.
- Lösung: komplette Trennung von Programmlogik (PHP) und Layout (HTML) mit Hilfe von Templates.
- Prominente Template-Engines:
 - Einfach: template.inc aus der PHPLIB
 - Umfangreicher Funktionsumfang: Smarty (<http://www.phpinsider.com/>)

HTML-Ausgabe via Template

- Funktionsweise:
 - Das Template besteht aus HTML-Code und Platzhaltern für einzufügende dynamische Daten.
 - Im PHP-Code wird kein HTML ausgegeben, sondern die Platzhalter werden durch spezielle Befehle der Template-Engine mit Daten belegt.
 - Die Template-Engine lädt auf Befehl das Template, ersetzt die Platzhalter mit den zugewiesenen Daten und gibt letztendlich die so produzierte fertige Webseite aus.



PHP-Entwicklungsumgebungen

- PHPMole (frei):

<http://www.akbkhomes.com/Projects/phpmole-IDE/>

- Zend Studio (kommerziell):

<http://www.zend.com/store/products/zend-studio.php>

- PHP Coder (frei):

<http://www.phpide.de/>

- Siehe auch in der PHP FAQ, Punkt 3.6:

<http://www.koehntopp.de/php/>