PHP 4

Einführung

Was ist PHP?

- PHP: Hypertext Processor
- offene Script-Sprache zur Erstellung dynamischer Webseiten
  - serverseitige Programmiersprache
  - Webseiten werden erst auf Anforderung des Clients erzeugt

Geschichte von PHP (1)

- Ende 1994
  - Personal Home Page Tools
  - eine Sammlung von Perl Skripten
- 1996
  - PHP/FI
    (Personal Homepage Tools/Form Interface)
  - erstmals Einbettung in HTML-Code
  - Umsetzung durch einen Interpreter

Geschichte von PHP (2)

- Sommer 1997
  - PHP 3
  - komplette Überarbeitung der Scripting-Engine
  - vollständige Analyse des Programm-Codes vor der Ausführung
  - langsam bei umfangreichen Web-Projekten
  - „execute while parsing“

Geschichte von PHP (3)

- Anfang/Mitte 2000
  - PHP 4
  - unabhängig vom Webserver
  - „compile first, execute later“

Warum PHP?

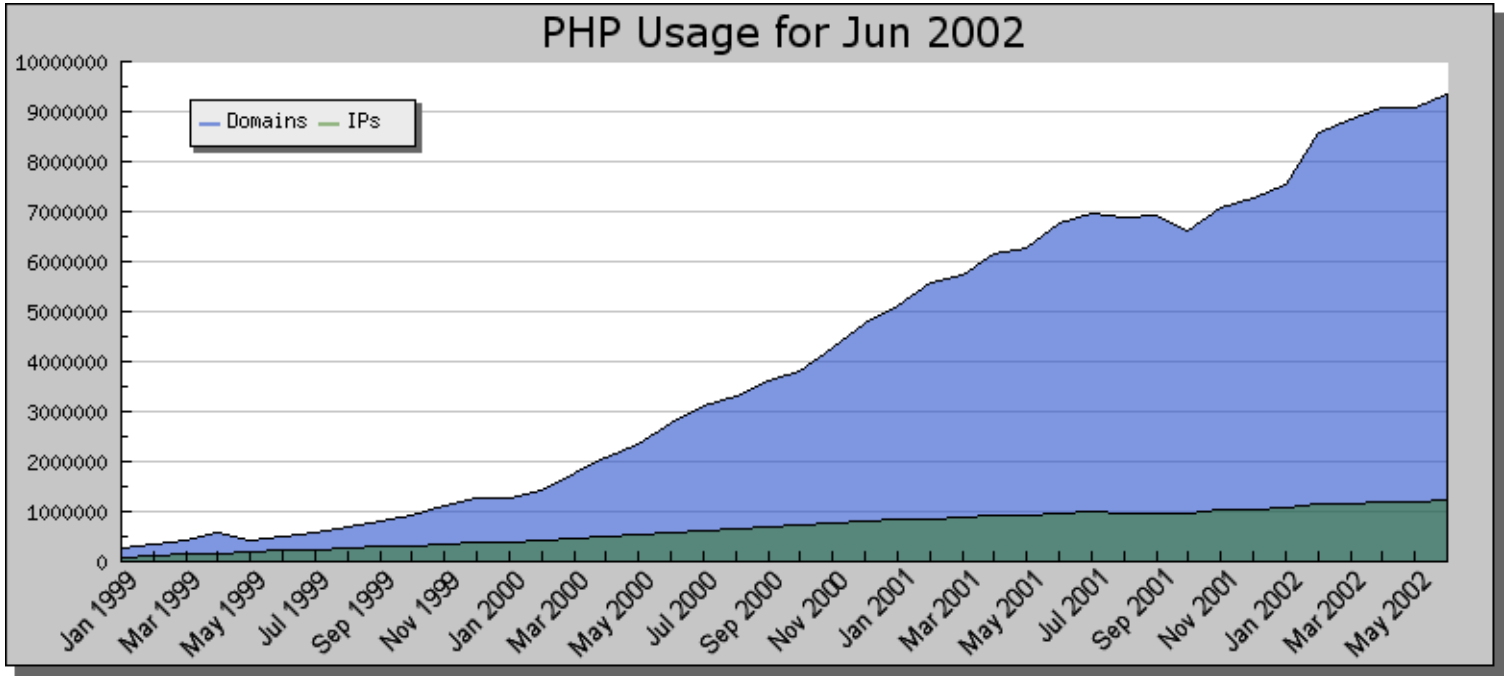
- Im Vorfeld nur HTML-Kenntnisse notwendig
- Open Source
- Abgrenzung zu anderen Techniken dynamischer Webseiten auf dem Server
  - CGI
  - ASP

Verbreitung von PHP (1)

- Open Source
- Webserver Unterstützung
  - Apache (Unix/Windows)
  - IIS / PWS
  - Netscape
  - → als SAPI, CGI und Servlet

Verbreitung von PHP (2)

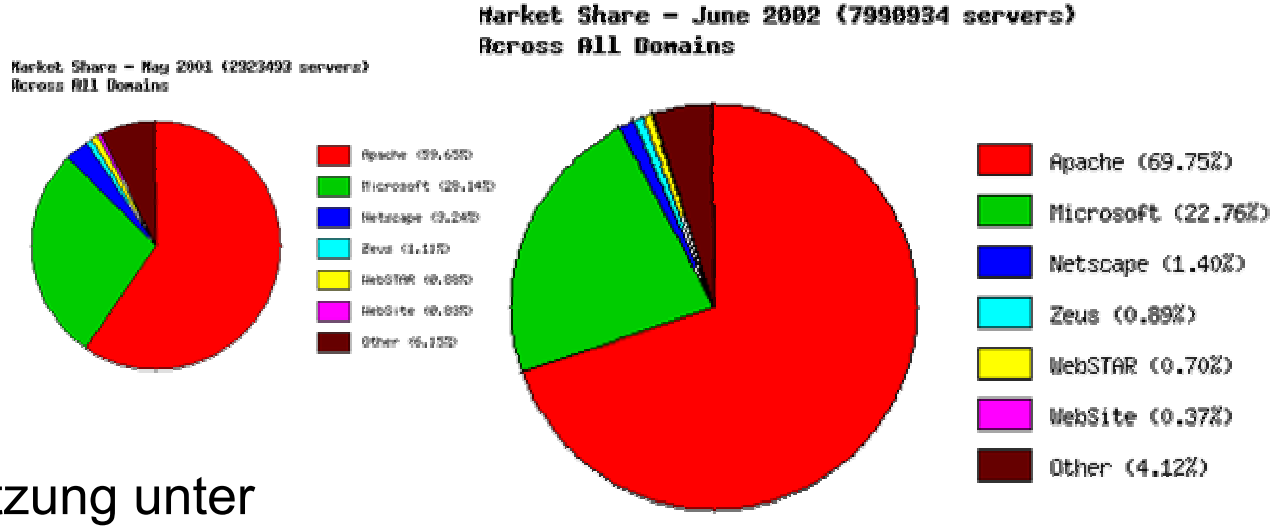
- PHP: 9,356,880 Domains,
  1,234,295 IP Addresses



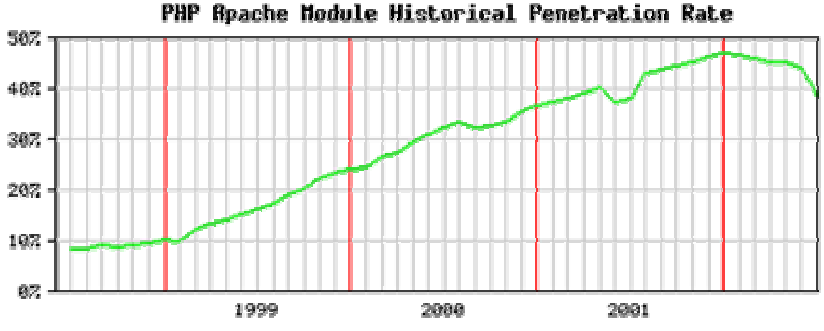
Quelle: Netcraft

Verbreitung von PHP (3)

- Webserver Verbreitung



- PHP Unterstützung unter Apache



Quelle: E-Soft WebSurvey

Abgrenzung zu CGI

- CGI Bewertung:
  - weit verbeitet
  - HTML geht im Programmcode unter
  - Out-Process-Anwendung
  - Skripts und kompilierte Programme
  - Programmierung oft zu aufwendig

→ PHP kann als CGI eingebunden werden

Abgrenzung zu ASP

- VBScript an die Bedürfnisse des Webservers angepasst
- Alternative zu ASP auf nicht Windows-Systemen
- keine Lizenzgebühren

Vorteile von PHP

- SAPI Unterstützung
- COM/DCOM Unterstützung
  unter Windows
- Session Management
- Verschlüsselung
  (Blowfish, TripleDES, MD5, SHA1)
- Datenbankanbindung
  (mySQL, MS-SQL, Oracle, …)

Vorteile von PHP

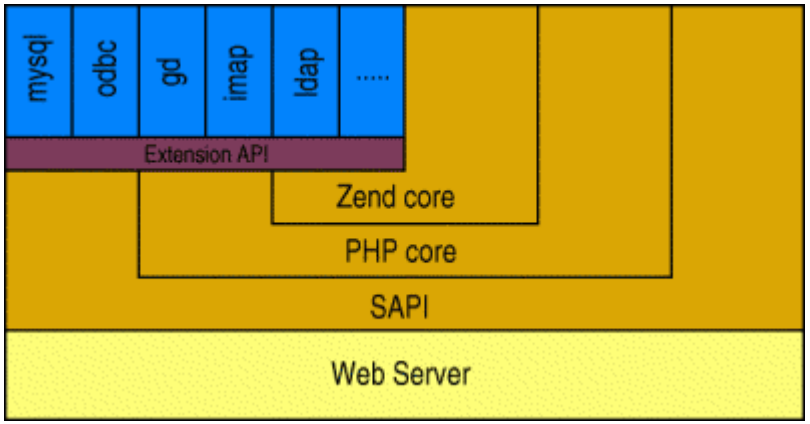
- FTP/LDAP/SMNP-Unterstützung
- Socket-Unterstützung
- Erzeugung von GIFs, Flash-Movies und PDF-Dokumenten
- Senden von HTTP-Header
- Bearbeiten von Zip-Dateien

Grundsätzliches zur Sprache

- Syntax an C und Perl angelehnt
  - Semikolons
  - Kommentare
- Typenlosigkeit
- Case Sensitivty
- „objektorientiert“
- sehr viele Funktionen

Ausblick

- Architektur von PHP



→ PHP als Skriptsprache für Applikationen

Weitere Spracheigenschaften

- Variablen beginnen mit „$“
```
$wert = "Hello";
```
- Zeichenketten werden mit „.“ verkettet
```
echo $wert . " World!";
echo "$wert World!";
```
- Funktionsaufrufe mit Referenzparameter
```
function mach_was (&$x)
```
- Inkrement/Dekrement-Operatoren

Beispiele

- Hello World

```
<body>
  <?
    echo "Hello World!";
  ?>
</body>
```

Programmkonstrukte

- Variablen
  - typenlos

```
$wert = "17";
```

  - Konvertierung
    - int, integer, string, double, real, array, object

```
(int) $wert;

Settype($wert, "int");

intval($wert);
```

Programmkonstrukte

- Variablen (Fortsetzung)
  - Datentypbestimmung

    ```
    gettype($wert)
    is_int($wert)
    ```

  - Existenz

    ```
    isset($wert)
    ```

  - Prüfung auf Wertzuweisung

    ```
    empty($wert)
    ```

  - Zuweisung aufheben

    ```
    unset($wert)
    ```

Programmkonstrukte

- Konstanten
  - Deklaration
    - Konvention: Großbuchstaben

```
define("WERT", 17);
```

  - Verwendung
    - ohne $

```
echo (WERT);
```

  - Konstante definiert

```
defined("WERT")
```

Programmkonstrukte

- Bedingungen
  - If-Bedinung

```
if (Bedingung) Anweisung;

if (Bedingung) {
  Anweisung;
  Anweisung;
}

if (Bedingung) {
  Anweisung;
} else {
  Anweisung;
}
```

```
if (Bedingung)
  Anweisung;
  Anweisung;
endif
```

Programmkonstrukte

- Bedingungen
  - Ternärer Ausdruck

```
Bedingung ? Anweisung1 : Anweisung2
```

  - Switch-Anweisung

```
switch($Variable) {
  case Bedingung1:
       Anweisung1;
       break;
  case Bedingung2:
       Anweisung2;
       break;
}
```

Programmkonstrukte

- Schleifen
  - While-Schleife (kopfgeprüft)

```
while (Bedingung) {
  Anweisung;
}
```

  - Do…While-Schleife (fußgeprüft)

```
do {
  Anweisung;
} while (Bedingung)
```

Programmkonstrukte

- Schleifen
  - for-Schleife

```
for(Start; Bedingung; Interation) {
  Anweisung;
}
```

  - foreach-Schleife

```
foreach(array as element) {
  Anweisung
}
```

Programmkonstrukte

- Funktionen
- Deklaration

```
function name($param1, $param2, …) {
  Anweisung;
  return $rueckgabewert;
}
```

- Nutzung

```
name($wert1, $wert2);
```

- variable Argumentenliste

```
function name($param1, $param2 = Wert)
```

Vergleich PHP vs. ASP

- Ausgabe der Zahlen 1 bis 10
  PHP

```
<body>
Hier kommen 10 Zahlen:<p>

<?
for ($i=1;$i<=10;$i++) {
   print "Zeile " . $i ."<br>";

}
?>


Das wars!<p>
</body>
```

- Ausgabe der Zahlen 1 bis 10
  ASP

```
<body>
Hier kommen 10 Zahlen:<p>

<%
 for i = 1 to 10
   Response.Write "Zeile " &
   i & "<br>"
 next
%>


Das wars!<p>
</body>
```

PHP 4

Formulare

Formularauswertung (1)

- PHP erkennt selbständig angehängte Daten
- Einbindung als normale Variable
  - Unabhängig von POST und GET
  - Manuelle Konfiguration notwendig (PHP.ini)

```
track_vars=on
```

Formularauswertung (2)

- Zugriff über die Arrays
  - $HTTP_GET_VARS
  - $HTTP_POST_VARS

```
echo $HTTP_POST_VARS["Eingabe"]
```

- Seit PHP 4.1.0
  - $_GET
  - $_POST

```
echo $_GET["Eingabe"]
```

Formularauswertung (3)

- Mehrwertige HTML-Felder

```
<input type="checkbox" name="checkboxfeld[]"
  value="Wert1">Wert1<br>
<input type="checkbox" name="checkboxfeld[]"
  value="Wert2">Wert2<br>
<input type="checkbox" name="checkboxfeld[]"
  value="Wert3">Wert3<br>
```

  - Name erfordert [] (Array)
- Bearbeitung in PHP

```
$arr= $_POST["checkboxfeld"];
for ($i=0; $i<count($arr); $i++)
  print($arr[$i]." ");
```

Datum und Uhrzeit

- Auslesen der Uhrzeit/Datum
  - Zeitstempel in Sekunden seit 1.01.1970
  - time()
- Formatieren von Uhrzeit/Datum
  - date("d.m.Y H:i:s", time())
  - date("d.m.Y H:i:s")

PHP 4

Session-Management

Cookies (1)

- Wert auslesen
  - Array mit Cookiewerten

```
$_COOKIE["Cookiename"];
```

- Wert setzen

```
setcookie("Cookiename", Wert, Gültigkeit)
```

- Wert löschen

```
setcookie("Cookiename", Wert, 0)
```

Cookies (2) - Beispiele

- Auslesen eines Cookies

```
$wert = $_COOKIE["counter"];
```

- Setzen eines Cookies
  - nur in der Browsersitzung

```
setcookie("counter", $wert);
```

  - 3 Minuten gültig

```
setcookie("counter", $wert, time()+180);
```

PHP Session-Management (1)

- Sessionfunktionalität
  - vergleichbar mit ASP
  - Nutzung
    - als "normale" Variable
      - Konfiguration erforderlich: `register_globals = On`
      - Registrierung der Sessionvariablen
    - über **`$_SESSION[]`** oder **`$HTTP_SESSION_VARS[]`**
      - keine Registrierung notwendig
  - Vorteil: Funktioniert auch ohne Cookies

PHP Session-Management (1)

- Start der Sitzung
  - Session-ID wird erzeugt
  - explizit

    ```
    session_start();
    ```
  - implizit durch Registrierung der Variablen

    ```
    session_register("counter");
    ```
  - Auslesen der Session-ID

    ```
    session_id()
    ```
- Beenden der Sitzung

  ```
  session_destroy();
  ```

PHP Session-Management (3)

- Beispiel:

```
<?
  session_start();
?>
<html><body>
<?
  if (!isset($_SESSION["count"])) {
    $_SESSION["count"] = 0;
  } else {
    $_SESSION["count"]++;
  }
  echo $_SESSION["count"]+1;
?>
</body></html>
```

PHP Session-Management (4)

- Ohne Cookies
  - Seitenwechsel durch Links/Formulare
  - Session-ID wird automatisch angehängt

```
echo ("<a href=\"" . $_SERVER['PHP_SELF'] .
  "\">weiter</a>");
```

PHP 4

Datenbank-Anbindung

Anbindung an Access (1)

- Nutzung von COM
  - Variablen belegen
  - Recordset erzeugen

```
$rs = new COM("ADODB.RecordSet");
```

  - Verbindung öffnen

```
$rs->open ($SQLQuery, $Conn, 3, 3);
```

  - Anzahl Datensätze

```
$num_columns = $rs->Fields->Count();
```

Anbindung an Access (2)

- Werte auslesen

```
echo $rs->Fields["SpaltenName"]->value;
```

- Zum nächsten Datensatz

```
$rs->movenext();
```

- Datenbank schliessen

```
$rs->Close();
$rs=null;
```

Anbindung an mySQL (1)

- Datenbankverbindung öffnen

```
$link=mysql_connect("mysql_host",
  "mysql_user", "mysql_password")
```

- Datentabelle wählen

```
mysql_select_db("my_database")
```

- SQL-Query

```
$query = "SELECT * FROM my_table";
$result = mysql_query($query) or
  die("Query failed");
```

Anbindung an mySQL (2)

- Datensätze ermitteln und ausgeben
  - Datensatz als assoziatives Array

```
while ($row = mysql_fetch_array($result,
  MYSQL_ASSOC)) {
        echo $row["Spaltenname"]
}
```

Anbindung an mySQL (3)

- Datensätze freigeben

```
mysql_free_result($result);
```

- Verbingung schliessen

```
mysql_close($link);
```