Manfred Kaul:
Database applications on the internet

RAD for the World Wide Web
www.php.net

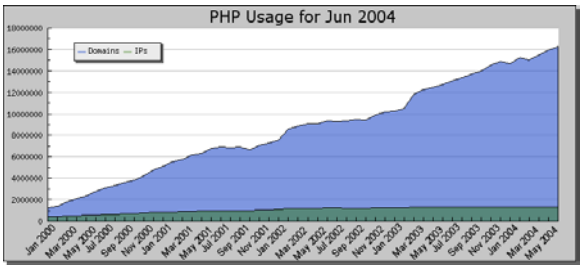
Contents

- History of PHP
  - Danish inventor Rasmus Lerdorf
  - German coordinator Egon Schmid
- PHP Language Basics
- Three-tier architecture
- Advantages
- Disadvantages

http://www.php.net/usage.php

PHP on the Internet

- **Netcraft Statistics**
  - **PHP: 16,251,453 Domains, 1,346,521 IP Addresses**



History of PHP

- PHP first version (originally known as Personal Home Pages) released in early 1995 by Rasmus Lerdorf
- 1995 : PHP/FI (Form Interpreter)
- 1997 : PHP3, renamed PHP: Hypertext Preprocessor - a recursive acronym.
- 1999 : PHP4
- 2003 : PHP5

- Over two hundred regular contributors

PHP: the Performance Champion

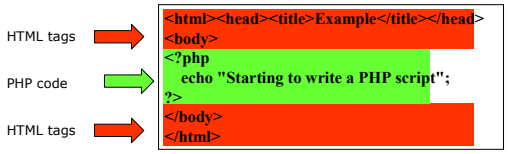
- From PHP HOWTO, July 2001
- Zdnet Statistics
  - PHP pumped out about 47 pages/second
  - Microsoft ASP pumped out about 43 pages/second
  - Allaire ColdFusion pumped out about 29 pages/second
  - Sun Java JSP pumped out about 13 pages/second

The Idea

- Write your HTML pages as usual
  - Embed PHP code as needed
    - Mark up code by special brackets
      - <?PHP ... ?>
  - Advantage: Easy startup
- ... or the other way around
  - Embed HTML into PHP code
    - multi-line strings no problem
    - embedded variable substitution
    - minimum amount of escape sequences

What is a PHP script ?

- Code embedded within tags : jumping between HTML and PHP (like ASP and Cold Fusion)
- No heavy code to output HTML (as needed in Perl or C)
- PHP code is enclosed in special start and end tags



PHP einbinden

- PHP wird in HTML-Dokumenten eingebunden, läuft nicht extern ab (wie Perl)
  - XML: <?PHP ... ?>
  - SGML: <? ... ?>
  - ASP: <% ... %>
  - <script language="PHP"></script>
  - PHP-Befehle müssen mit Semikolon abgeschlossen werden

PHP script with $Variables

- When you call PHP from an HTML form, the form fields are turned into $variables.

```
<?php
  $Today = date("l F d, Y");
?>
<html>
<head>
<title>Listing 1-10</title>
</head>
<body>
Today's date:
<?php
   /*
   ** print today's date
   */
```

```
print("<h3>$Today</h3>\n");

   /*
   ** print message about lunch cost
   */
   print($_REQUEST['YourName'] . ", you will be out ");
   print($_REQUEST['CostOfLunch'] *
      $_REQUEST['DaysBuyingLunch']);
   print(" dollars this week.<br>\n");
?>
</body>
</html>
```


$Variables

- To use or assign variable **$** must be present before the name of the variable
- The **assign** operator is '='
- There is **no need to declare the type** of the variable
- the current stored value produces an **implicit type-casting** of the variable.
- A variable can be used before to be assigned

```
$A = 1;
$B = "2";
$C = ($A + $B); // Integer sum
$D = $A . $B; // String concatenation
echo $C; // prints 3
echo $D;//  prints 12
```


Variable Variables

- Function isset tests if a variable is assigned or not

```
$A = 1;
if (isset($A))
        print "A isset"
  unset($A);
if (!isset($A))
        print "A is NOT set";
```

dictionary of variables

- Using $$

```
$help = "hiddenVar";
$$help = "hidden Value";  // $hiddenVar = "hidden Value";
echo $$help; // prints  hidden Value
$$help = 10;
$help = $$help * $$help;
echo $help; // print 100
```


Conditional

```
<html>
<head>
<title>Listing 1-11</title>
</head>
<body>
<h1>
<?php
   $Today = date("l");
   if($Today == "Friday")
   {
      print("Thank goodness it's Friday!");
   }
   else
   {
      print("Today is $Today.");
   }
?>
</h1>
</body>
</html>
```

- PHP allows you to test conditions and execute certain code based on the result of the test.
- The simplest form of this is the if statement.

Repeating

```
<html>
<head>
<title>Listing 1-12</title>
</head>
<body>
<h1>Today's Daily Affirmation</h1>
Repeat three times:<br>
<?php
   for($count = 1; $count <= 3; $count++)
   {
      print("<b>$count</b> I'm good enough, ");
      print("I'm smart enough, ");
      print("and, doggone it, people like me!<br>\n");
   }
?>
</h1>
</body>
</html>
```

- The last type of functionality in this brief introduction is looping.
- Looping allows you to repeat the execution of code.

PHP Language Basics

- **Hello World!: An Example (cont.)**
  - <HTML>Hello World</HTML>
  - <?PHP
    echo 'Hello World';
    ?>
  - <script language="PHP">
    $hello = "Hello";
    $world = "World!";
    print $hello . $world;
    </script>

PHP Language Basics

- **Constants, Data Types and Variables**
  - Constants define a string or numeric value
  - Constants do not begin with a dollar sign
  - Examples:
    - define('COMPANY', 'Acme Enterprises');
    - define('YELLOW', '#FFFF00');
    - define('PI', 3.14);
    - define("HOME", "C:"); define("HOME", "H:");
  - Constants must be defined only once

PHP Language Basics

- **Constants, Data Types and Variables**
  - Using a constant
    - print("Company name: " . COMPANY . NL);

PHP Language Basics

- **Constants, Data Types and Variables**
  - Data types
    - Integers, doubles and strings
      - isValid = true; // Boolean
      - 25 // Integer
      - 3.14 // Double
      - 'Four' // String
      - "Total value" // Another string

PHP Language Basics

- **Constants, Data Types and Variables**
  - Data types
    - Strings and type conversion
      - $street = 123;
      - $street = $street . " Main Street";
      - $city = 'Naperville';
        $state = 'IL';
      - $address = $street;
      - $address = $address . NL . "$city, $state";
      - $number = $address + 1; // $number equals 124

PHP Language Basics

- **Constants, Data Types and Variables**
  - Data types
    - Arrays
      - Perl-like syntax
        - $arr = array("foo" => "bar", 12 => true);
      - same as
        - $arr["foo"] = "bar";
        - $arr[12] = true;

PHP Language Basics

- **Constants, Data Types and Variables**
  - Arrays (cont.)

```
<?php
    $arr = array("somearray" => array(6 => 5, 13 => 9,
                 "a" => 42));

    echo $arr["somearray"][6];    // 5
    echo $arr["somearray"][13];   // 9
    echo $arr["somearray"]["a"];  // 42
?>
```


PHP Language Basics

- **Constants, Data Types and Variables**
  - Operators
    - Contains all of the operators like in C and Perl (even the ternary)
  - Statements
    - if, if/elseif
    - Switch/case
    - for, while, and do/while loops
    - Include and require statements for code reuse

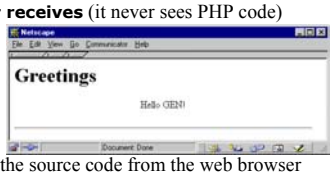
PHP is server-side

What the web server sees (and executes on-the-fly)

```
<H1>Greetings</H1>
<P ALIGN="center">
<?php
print "Hello GEN!";
?>
</P><HR>
```

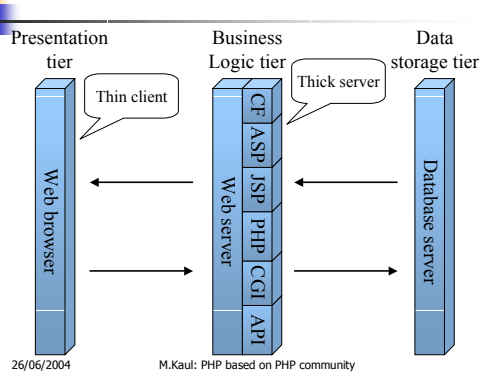
What the web browser receives (it never sees PHP code)

```
<H1>Greetings</H1>
<P ALIGN="center">
Hello GEN!
</P><HR>
```

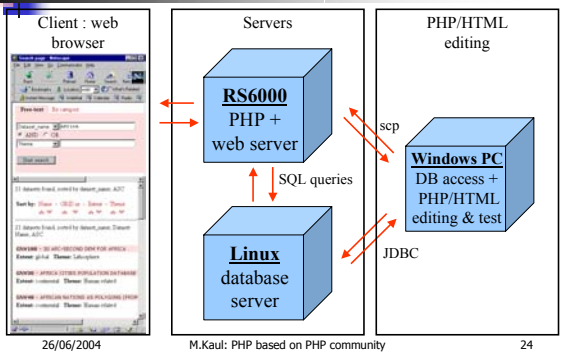


=> it is not possible to copy the source code from the web browser

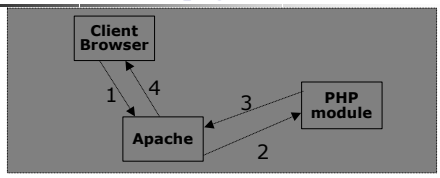
Three-tier architecture



Architecture : an example



How PHP generates HTML/JS Web pages



1: Client from browser send HTTP request (with POST/GET variables)
2: Apache recognizes that a PHP script is requested and sends the request to PHP module
3: PHP interpreter executes PHP script, collects script output and sends it back
4: Apache replies to client using the PHP script output as HTML output

What do you need to use PHP ?

- PHP in one of the following flavours:
  - executable : stand-alone (PHP-GTK), CGI (php.exe)
  - module : Apache web server (php4apache.dll), ISAPI (php4isapi.dll, not in production state)
- A web server
- A database server
- Optional :
  - extensions : additional functionalities
  - an ODBC database driver for accessing to/from other databases

System requirements

- Operating systems :Linux, many Unix variants (HP-UX, Solaris and OpenBSD, ...), Microsoft Windows, Mac OS X, RISC OS, …
- Web servers : Apache, Microsoft Internet Information Server, Personal Web Server, Netscape, iPlanet, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd, Zeus, ...
- Databases : Oracle, Informix, Sybase, InterBase, PostgreSQL, MySQL, ODBC, ...

How to install PHP ?

=> http://www.zend.com/manual/installation.php

- Let's have the Open Source attitude : compile the program from the source code
- Get the executable binaries
- Get PHP with your operating system (ex: RedHat Linux)
- Use an installer
  - PHP installer at http://www.php.net/
  - PHP + Apache + mysql installer : EasyPHP.org
- Call your beloved system manager

Apache + PHP + MySQL complete

Installation for Windows

- Get EasyPHP 1.5 (9 MB) at http://www.easyphp.org/
- Double-click on easyphp1-5_setup.exe
- Use myphpadmin to manage your mysql database
- Install the myodbc driver (http://www.mysql.com/downloads/api-myodbc.html)
- Create a new ODBC data source
- Access the data source with MS-Access

Write a PHP file

- Buy a good PHP manual, or go to http://www.php.net/manual/
- Open your favorite text editor
- Type your PHP/HTML/Javascript code
- Save your file with the extension .php
- Put your .php file on your (local) web server
- Access your .php file with a web browser using the http protocol:
  - http://localhost/my_file.php
- Do not use the following location
  - file://d:/Program%20Files/EasyPHP/www/my_file.php

include und require

- Zentrale Funktionen oder wiederkehrende Struktur-elemente per include verwalten
- include("datei.typ");
- include_once "datei.typ" ;
- require("datei.typ");
- require_once "datei.typ" ;

dynamic

static

```
if ( $x = 0) for ($i=0; $i<10; $i++) include("test.txt");
                                     require
```


Kommentare

- Einzeilige Kommentare:
  - // Dies ist ein Kommentar
- Mehrzeilige Kommentare:
  - /* Dies ist auch ein Kommentar, nur ein mehrzeiliger */

Variablen und Konstanten

- $DEBUG = 0 ;
- $varname = "DEBUG";
- echo ${$varname};
- for ($i=0; $i<10; $i++){
  $$i = $i;
  echo $$i . "
";
  }
- $a[2]="value at 2";
- print_r($a);

- define("DEBUG" , 0);
- define("DEBUG" , 0);
- echo DEBUG;

Variablen und Arrays

- Keine statische Typisierung
- **Dynamische** Typisierung
  - Typ muss nicht festgelegt werden, ergibt sich dynamisch durch Zuweisung
- Arrays: $myArray[] = "Hans";
  $myArray[] = "Otto";
  $myArray[] = 3; Heterogene Typen
- $myArray = array("Hans", "Otto", 3);
- echo myArray[1] // ergibt "Otto"
- mapArray["vorname"]="Otto";

Assoziative Arrays

- $arr = array("a" => "apple", "b" => "banana");
- **reset** ($arr);
  while (list($key, $value) = each ($arr)) {
  echo "Key: $key; Value: $value
\n";
  }
- **foreach**($arr as $key => $value) {
  print "\$arr[$key] => $value.\n";
  }

Ersetzung

- echo "Hallo $vorname $nachame !!!";
- echo "Hallo". $vorname." ".$nachame . "!!!";
- **printf**('Hallo %s %s !!!', $vorname, $nachname);
- echo **sprintf** ('Hallo %s %s !!!', $vorname, $nachname);

Existenz von Variablen

- Prüfen ob die Variable existiert: isset($var);
- oder nicht: !isset($var);
- Variable leer: empty($var);
- oder nicht: !empty($var);
- Variable uninitialisieren: unset($var);

Operatoren

- $var0==$var1
- $var0!=$var1
- <,>,<=,>=
- && bzw. ||
- +,-,*,/
  - echo 3/5; // ergibt 0.6
- $var++, $var--
- und viele andere ...

Funktionen

```
function hello1( $vorname, $nachname ){
        echo "hello1> Hello $vorname " .
          " $nachname";
}

hello1( "Otto", "Müller" );
```


Default-Parameter

```
function hello2( $vorname, $nachname="Müller"  ){
    echo "hello2> Hello $vorname $nachname";
}

hello2( "Otto" );
hello2( "Otto", $nachname="Maier" );
```


Parameter by &reference

```
function set_var( &$var, $value ){  // by reference
    $var = $value;
}
$var1 = 1;
set_var( $var1, 2 );
echo $var1;  // ergibt 2
function set_global_var( $value ){
    global $var1;
    $var1 = $value;
}
set_global_var( 3 );
echo $var1;  // ergibt 3
```


Funktionen als Parameter

```
function hello3( $fun, $vorname, $nachname ){
      echo "hello3>";
      $fun( $vorname, $nachname );
  }

hello3( "hello1", "Otto", "Müller" );
```

Funktionale Programmierung in PHP

```
function map( $fun, $array ){
  foreach($array as $key => $value)
      $result[$key] = $fun($value);
  return $result;
}

function quadrat( $x ){ return $x * $x; }
print_r( map( "quadrat", array(1,2,3,4) ));
```

Wende eine Funktion auf alle Elemente eines Arrays an.

Funktionale Programmierung in PHP

- **array_map** (mixed callback, array arr1 [, array ...])
  - Applies the callback to the elements of the given arrays
- **array_filter** (array input [, callback function])
  - Filters elements of an array using a callback function
- **array_reduce** (array input, callback function [, int initial])
  - Iteratively reduce the array to a single value using a callback function

Reguläre Ausdrücke

- Test, ob Wortteile oder Aufbau von Zeichenketten einer bestimmten Struktur entsprechen
- ereg("*exp*", $var)
  - if (eregi("z", $string)) {
    echo "'$string' contains a 'z' or 'Z'!";
    }
- eregi, ereg_replace, eregi_replace
  - A Quick way of removing excess spaces:

```
<?
$string = "One        Two         Three   Four";
$var = eregi_replace(" +", " ", $string);
echo $var;
?>
```


Klassen in PHP4

- Schlüsselwörter
  - **class** Klassenname
  - **var** Instanzenvariable
  - **function** Instanzenmethode
  - **$this** Referenz auf die Instanz selbst
- PHP3 kannte keine Klassenvariablen und Klassenmethoden
- PHP5 hat Klassen und Objekte wesentlich optimiert

Cart.class.php

```
class Cart {
      var $items;          // Instanzen-Variable: Items in our shopping cart

      // Add $num articles of $artnr to the cart
      function add_item ($artnr, $num){    // Instanzenmethode
        if ( ! array_key_exists($artnr, $this->items) )
                $this->items[$artnr] = $num;
        else
                $this->items[$artnr] += $num;
      }
}
$cart = new Cart;                          // Instanziierung
$cart->add_item("10", 1);                  // Methoden-Aufruf
```


Konstruktor

```
class Cart {
      var $items;  // Items in our shopping cart

      function Cart(){
         $this->items = array();
      }

      // ...
}
$cart = new Cart();
```

Konstruktor

Konstruktoren / Destruktoren

- Keine komplexe Speicherverwaltung
  - PHP-Skriptausführung von kurzer Dauer
  - Aufräumen kostet Zeit
- Konstruktor optional
- Destruktoren nicht vorhanden
- Speicherplatz wird durch new allokiert
- Konstruktor initialisiert Objektvariablen

Beispiele Instanziierungen

- $cart1 = new Cart;
  - Instanzenvariablen werden nicht initialisiert (leer)
  - Kein Konstruktor-Aufruf
- $cart2 = new Cart();
- $cart3 = new Cart("BMW", 3);
  - Explizite Initialisierung der Instanzenvariablen durch Konstruktor-Aufruf

Referenzierung

- Zugriff auf Objektvariablen und Objektfunktionen
  - echo $cart->items; ($ vor items entfällt!!)
  - $cart->ausgeben();
- „Statischer" Funktionsaufruf
  - Cart::funktion();
- Fehlerausgabe bei unbekannter Funktion

Vererbung mit **extends**

- Vererbung
- Einer der Grundgedanken der OOP
  - Oberklasse → Unterklasse
- Schlüsselwort: **extends**
  - class LKW **extends** Auto
- Im Gegensatz zu C++ keine Mehrfachvererbung

Reflection

```
$object[0] = new LKW("M.A.N","orange", 5000);
$object[1] = new Auto("Volkswagen","grün");
for ($i=0; $i<count($object); $i++) {
  if (get_class($object[$i]) == „Auto") {
     print("Objekt ist ein Auto<br>");
  } else {
     print("Objekt ist ein LKW<br>");
  }
}
```


Reflection API -1-

- class_exists -- Checks if the class has been defined
- get_class -- Returns the name of the class of an object
- get_class_vars -- Returns an array of default properties of the class
- get_class_methods -- Returns an array of class methods' names
- call_user_method_array -- Call a user method given with an array of parameters [deprecated]
- call_user_method -- Call a user method on an specific object [deprecated]

Reflection API -2-

- get_declared_classes -- Returns an array with the name of the defined classes
- get_object_vars -- Returns an associative array of object properties
- get_parent_class -- Retrieves the parent class name for object or class
- is_a -- Returns TRUE if the object is of this class or has this class as one of its parents
- is_subclass_of -- Returns TRUE if the object has this class as one of its parents
- method_exists -- Checks if the class method exists

Now… Let's do SQL on the web

PHP style

Datenbank-Anbindung

- Extensions für viele Datenbank
  - MySQL
  - Oracle, db2
  - PostgreSQL
  - Adabas
  - Sybase
  - Interbase
  - filePro
  - Ovrimos ... .

Schon freigeschaltet

php.ini
;extension=pgsql.dll oder .so

PostgreSQL-Anbindung

```
$PG_HOST     = 'ux-2d01.inf.fh-brs.de';
$PG_PORT     = "5432";
$PG_DB       = "bibdb";
$PG_USER     = "bibdbuser";
$PG_PASSWORD = "...";
$connection_string  = "";
$connection_string .= ' host=' . $PG_HOST;
$connection_string .= ' port=' . $PG_PORT;
$connection_string .= ' dbname=' . $PG_DB;
$connection_string .= ' user=' . $PG_USER;
$connection_string .= ' password=' . $PG_PASSWORD;
$pg_dbh = pg_connect( $connection_string );
```


PostgreSQL-Anbindung

```
$pg_dbh = pg_connect( $connection_string );
  if ( ! $pg_dbh ) die( "<H1>Failed to connect to
   $PG_HOST </H1>" );

     $sql  = " SELECT *   ";
     $sql .= " FROM Namen ";
or
     $sql  = sprintf( "SELECT ...%s %d", $t, $d );
```

multiline strings allowed

PostgreSQL-Anbindung

```
$pg_result = pg_exec($pg_dbh, $sql);

   if ( $pg_result ){
      $num = pg_num_rows($pg_result);
      for ($i=0; $i < $num; $i++) {
         $row = pg_fetch_row($pg_result, $i);
         list($Name, $id) = $row;
         echo "Name $Name hat id $id.<BR>";
      }
```

_ entfällt in älteren Versionen

PostgreSQL-Anbindung

```
$pg_result = pg_exec($pg_dbh, $sql);

    if ( $pg_result ){
        $num = pg_num_rows($pg_result);
        for ($i=0; $i < $num; $i++) {
            $row = pg_fetch_row($pg_result, $i);
            list($Name, $id) = $row;
            echo "Name $Name hat id $id.<BR>";
        }
    } else {
        echo pg_last_error();
    }
```


Basic PHP/SQL interaction

1. Open a connection to the database
2. If ok, generate SQL command
3. "Execute" SQL command
4. Handle responses from the server
5. If not done, go back to step 2
6. If done, close connection to database

Creating a table with PHP

```
<?php
// create a connection with the database
$connection=pg_Connect("host=ux-2d01.inf.fh-brs.de port=5432
    dbname=mgmt357 user=mgmt357 password=ecommerce");
// if there is no connection, generate an error and get out
if (!$connection) {
    print pg_ErrorMessage(); exit(-1);
};
// Create the table with the pg_exec command
$result=pg_exec($connection,"create table customer (
                            id int8,
                            name varchar(50),
                            address varchar(50),
                            email varchar(64)
                    )");
if ($result) {
    print "The customer table has been created.";
} else {
    print pg_ErrorMessage();
}
pg_close($connection);
?>
```

SQL

Entering data with PHP

```
<?php

if ($submit) {
 // $connection = ...

$id=time();  // set a unique id number to
    each record

$result=pg_exec($connection, "INSERT
    into customer VALUES
    ($id,'$name','$address','$email')");

if ($result) {
    print "The customer data has been
    inserted.<p>";
} else {
    print pg_ErrorMessage();
}
    pg_close($connection);
} else {
    // display form
?>

<form method="post"
    action="<?php echo $PHP_SELF?>">
Name: <input type="Text"
    name="name">
Address: <input type="Text"
    name="address">
Email:<input type="Text" name="email">
<input type="Submit" name="submit"
    value="Enter information">
</form>

<?
}
?>
```

mixture of HTML and PHP

HTML form

Updating data with PHP

```
<?php

if ($submit) {
// create a connection with the database
// ... as usual ...

$result=pg_exec($connection, "UPDATE
    customer
            set name='$name',
            set address='$address',
            set email='$email' where
    id='$id'");

if ($result) {
    print "The customer data has been
    inserted.<p>";
} else {
    print pg_ErrorMessage();
}
    pg_close($connection);
} else {
    // display form
?>

<form method="post"
    action="<?php echo $PHP_SELF?>">
Id: <input type="Text" name="id">
Name: <input type="Text"
    name="name">
Address: <input type="Text"
    name="address">
Email:<input type="Text" name="email">
<input type="Submit" name="submit"
    value="Enter information">
</form>

<?
}
?>
```


Displaying data with PHP

```
$result=pg_exec($connection,"select * from customer");
if ($result) {  // get the number of rows and store it in $r
    $r = pg_numrows($result);
    print "The customer table has $r row(s) of data.<p>";
    $row=0; // set the row counter to the beginning 0
    print "<table border=1>\n";
    // while there are records to deal with...
    while ($data=pg_fetch_object($result,$row)) {
        print "<tr><td>$data->name</td>";
        print "<td>$data->address</td>";
        print "<td>$data->email</td></tr>\n";
        $row++;          // increment the counter
    };
    print "</table>\n";
} else {
    print pg_ErrorMessage();
}
```

Generating HTML

Getting even easier… PEAR

- Abstracting the database interface

```
<?php
include('dbinfo.php');
require_once( 'DB.php' );
$db = DB::connect( "mysql://$user:$pass@$host/$dbname" );
// no need to select DB
$sql = 'SELECT * FROM demo';
$demoResult = $db->query($sql);
while ($demoRow = $demoResult->fetchRow()) {
    echo $demoRow[2] . '<br>';
}
$db->disconnect();
?>
```

http://www.phpbuilder.com/columns/allan20010115.php3?page=1

Advantages of PHP

- Modest learning curve
- Free and open development (independence, security, community)
- Native database connectivity (mysql, ODBC, …)
- Availability for a variety of platforms (Windows, UNIX, Linux, Mac OS, RISC)
- Simple but powerful, thin client
- Many functions (databases, mail server, PDF, …)

Disadvantages of PHP

- Immature language
- Mixing of HTML and program code : disorganization (application and layout not always well separated), bug prone
- Debugging complex errors with limited support
- In general, more efficient as an Apache web server module => CGI and other web servers' versions are often weaker (speed, reliability)

Resources

- PHP Sites
  - http://www.php.net
  - http://www.phpbuilder.com
  - http://www.devshed.com
  - http://www.geneseo.edu/~kma/PHP_Intro
- Books
  - *SQL in a Nutshell,* Kevin Kline, O'Reilly
  - *PostgreSQL: Introduction and Concepts,* Bruce Momjian, Addision Wesley
  - Introduction to Database Systems, C.J. Date