Database Systems

PHP & SQL

Departamento de Engenharia Informática
Instituto Superior Técnico

1st Semester
2009/2010

Outline

Outline

1. PHP
   - The PHP Language
   - Language Constructs
   - Arrays
   - Conditional Statements
   - Loops
   - Changing Pages
2. PHP and PostgreSQL

PHP

- PHP: **P**HP **H**ypertext **P**reprocessor
- Server-side scripting language
- Designed for the dynamic generation of Web pages
- Runs on many platforms (Linux, Mac, Windows, ...)
- Integrated with many Web servers (Apache, IIS, ...)
- Supports many databases (PostgreSQL, Oracle, ...)
  - Well integrated with PostgreSQL
- It's Open Source and Free Software (FSF-approved)

How PHP Works

- Installed with a Web server and a DMBS
  - Usually Apache and MySQL/PostgreSQL
- When the browser is pointed to a PHP file (.php)
  1. The server processes the file and sends any HTML statements
  2. When the server reaches a PHP tag (<?php) it executes the PHP statements and sends the resulting output
  3. When an ending tag is reached (?>), the server switches back to sending HTML statements

A simple program


```
<html>
    <head>
        <title>PHP Test</title>
    </head>
    <body>
        <?php echo '<p>Hello World</p>'; ?>
    </body>
</html>
```

(test me here)

Variables

- No need to declare variables, just assign a value

```
$age = 12;
$price = 2.55;
$number=-2;
$name = "Goliath Smith";
```

- Variable names:
  - start with $
  - can include only letters, numbers and underscore
  - cannot begin with a number
- Clearing variables

```
unset($age);
```

- Note: PHP is case-sensitive

Constants

- Constants are set using the `define` statement

```
define("COMPANY","ABC Pet Store");
define("AGE",29);

echo COMPANY;
echo AGE;
```

Operators

- PHP can perform arithmentic operations
  - $+$, $-$, $*$, $/$, $\%$
- Example:

```
$result = (1 + 2) * 4 + 1;
```

- The following comparison operators are available:
  - $==$, $>$, $<$, $>=$, $<=$, $!=$ (or $<>$)
- Example:

```
$weather == "raining"
$age < 13
```

- Plus the logical operators
  - and (or &&), or (or ||), xor and !

Strings

- A string:

```
$string = 'Hello World!';
```

- Quote characters (" and ') must be escaped

```
$string = 'It is Tom\'s house';
```

- String concatenation:

```
$string1 = 'Hello';
$string2 = 'World!';
$stringall = $string1 . ' ' . $string2;
```

- Formatting strings:

```
$price = 25;
$fprice = sprintf("%01.2f", $price);
```

Single-Quotes versus Double-Quotes

- Single-quoted strings (using ') are represented as is

```
$age = 12;
echo 'The age is $age';
```

yields

```
The age is $age
```

- Double-quoted strings (using ") are processed

```
$age = 12;
echo "The age is $age";
```

yields

```
The age is 12
```

- Special characters are also recognized
  - \n, \t, ...

Dates and Times

- Getting the current date and time

```
$today = time();
```

- Formatting a date

```
$cdate = date("d/m/y", $today);
$ctime = date("G:i:s", $today);
```

  - Many formatting options:
    - M: month abbreviated
    - F: month not abbreviated
    - ...

- Converting strings to dates

```
$importantDate = strtotime("January 15 2003");
```

(test me here)

Date Operations

- PHP can perform date arithmetic

```
$timeSpan = $today - $importantDate;
```

returns the number of seconds between both dates

- Conversion from strings also accepts many operations

```
$importantDate = strtotime("tomorrow");
$importantDate = strtotime("now + 24 hours");
$importantDate = strtotime("last saturday");
$importantDate = strtotime("8pm + 3 days");
$importantDate = strtotime("2 weeks ago");
$importantDate = strtotime("next year gmt");
$importantDate = strtotime("this 4am");
```

Regular Expressions

- PHP supports regular expressions

```
ereg("pattern", string);
```

returns true if *pattern* matches *string*

- Example:

```
ereg("^[A-Za-z' -]+$", $name)
```

checks the validity of a name

- Replacements:

```
$new = ereg_replace($expression,$replacement,$old)
```

Other Useful Statements

- PHP contains self-referring arithmetic operators operators

```
$counter += 2;
$counter -= 3;
$counter *= 2;
$counter /= 3;
```

- Exiting a PHP program

```
exit("The program is exiting");
```

or

```
die("The program is dying");
```

- Comments can be inserted using /* ... */ or #

Functions


```
function add_2_numbers($num1 = 1, $num2 = 1)
{
  $total = $num1 + $num2;
  return $total;
}
```

Creating Arrays

- Simple array

```
$pets[1] = "cat";
$pets[2] = "dog";
$pets[3] = "monkey";
```

or

```
$pets = array("cat","dog","monkey");
```

- Key-indexed array

```
$capitals['PT'] = "Lisbon";
$capitals['AU'] = "Canberra";
$capitals['BR'] = "Brasilia";
```

or

```
$capitals = array("PT" => "Lisbon",
   "AU" => "Canberra", "BR" => "Brasilia");
```

Sorting Arrays

- Sorting simple arrays

```
sort($pets);
```

- Sorting key-indexed arrays

```
asort($capitals);
```

- Other sort methods:
  - rsort, arsort: reversed sort
  - ksort, krsort: sort by key
  - usort($array,function): sort using *function*

Moving Through an Array

- Arrays can behave as iterators

```
reset($capitals);

$value = current($capitals);
echo "$value<br>";

$value = next($capitals);
echo "$value<br>";

$value = next($capitals);
echo "$value<br>";
```

- Other statements:
  - previous, end, sizeof

Moving Through an Array (cont.)

- Can use the foreach statement

```
$capitals = array ( "PT" => "Lisbon",
        "AU" => "Canberra", "BR" => "Brasilia" );
ksort ($capitals);
foreach ( $capitals as $state => $city )
{
  echo "$city, $state<br>";
}
```

(test me here)

Multidimensional Arrays

- Creating a multidimensional array

```
$productPrices['clothing']['shirt'] = 20.00;
$productPrices['clothing']['pants'] = 22.50;
$productPrices['linens']['blanket'] = 25.00;
$productPrices['linens']['bedspread'] = 50.00;
$productPrices['furniture']['lamp'] = 44.00;
$productPrices['furniture']['rug'] = 75.00;
```

- Using multidimensional arrays

```
$shirtPrice = $productPrices['clothing']['shirt'];
```

Multidimensional Arrays (cont.)


```
<?php
$productPrices['clothing']['shirt'] = 20.00;
$productPrices['clothing']['pants'] = 22.50;
$productPrices['linens']['blanket'] = 25.00;
$productPrices['linens']['bedspread'] = 50.00;
$productPrices['furniture']['lamp'] = 44.00;
$productPrices['furniture']['rug'] = 75.00;

echo "<table border=1>";
foreach( $productPrices as $category ) {
  foreach( $category as $product => $price ) {
    $f_price = sprintf("%01.2f", $price);
    echo "<tr><td>$product:</td><td>\$$f_price</td></tr>";
  }
}
echo "</table>";
?>
```

(test me here)

The *if* Statement


```
if ($country == "Germany" )
{
  $version = "German";
  $message = " Sie sehen unseren Katalog auf Deutsch";
}
elseif ($country == "France" )
{
  $version = "French";
  $message = " Vous verrez notre catalogue en francais";
}
else
{
  $version = "English";
  $message = "You will see our catalog in English";
}
echo "$message<br>";
```

The *switch* Statement


```
switch ($custCountry)
{
  case "PT" :
    $salestaxrate = 0;
    break;
  case "AU" :
    $salestaxrate = 1.0;
    break;
  default:
    $salestaxrate = .5;
    break;
}
$salestax = $orderTotalCost * $salestaxrate;
```

The *for* Loop


```
for ($i = 0; $i < sizeof($customerNames); $i++)
{
  echo "$customerNames[$i]<br>";
}
```

```
for ($i = 0, $j = 1;$t <= 4; $i++, $j++)
{
  $t = $i + $j;
  echo "$t<br>";
}
```

The *while* Loop


```
while ($testvar != "yes")
{
  if ($customers[$k] == "Smith")
  {
    $testvar = "yes";
    echo "Smith<br>";
  }
  else
  {
    echo "$customers[$k], not Smith<br>";
  }
  $k++;
}
```

The *do...while* Loop


```
do
{
  if ($customers[$k] == "Smith")
  {
    $testvar = "yes";
    echo "Smith<br>";
  }
  else
  {
    echo "$customers[$k], not Smith<br>";
  }
  $k++;
} while ($testvar != "yes")
```

- Note: continue and break statements can be used

Redirecting the Browser

- The PHP header function can be used to send a new page to the browser

```
if ($customer_age < 13)
{
  header("Location: ToyCatalog.php");
}
else
{
  header("Location: ElectronicsCatalog.php");
}
```

Note: the header function must appear before any output is sent

Using Cookies

- Setting cookies

```
setcookie("country","PT");
```

or

```
setcookie("country","PT",time()+3600);
```

- All values stored as cookies are available in the `$_COOKIE` array

```
echo $_COOKIE['country'];
```

Note: the *setcookie* function must be used before any output is sent

PHP Sessions

- Cookies are not always available
- PHP provides a safer mechanism: sessions
- When a session is started, PHP
  - If the session exists, uses it, if not creates a new session number
  - Stores session variables in a file, on the server
  - Passes the session id number to every page
    - through cookies, the URL, or a hidden post variable
  - For each new page, gets the variables from thew session file
    - The variables are accessible in the `$_SESSION` array

Using a Session

- To start a session

```
session_start()
```

Note: *session_start* must be used before any output is sent

- Using session variables

```
$_SESSION['country'] = "PT";
```

and, on a different page

```
echo $_SESSION['country'];
```

- Closing the session

```
session_destroy();
```

Session Example


```
<?php
  session_start();
?>
<html>
<head><title>Testing Sessions page 1</title></head>
<body>
<?php
  $_SESSION['session_var'] = "testing";
  echo "This is a test of the sessions feature.
       <form action='sessionTest2.php' method='POST'>
       <input type='hidden' name='form_var'
              value='testing'>
       <input type='submit' value='go to next page'>
       </form>";
?>
</body></html>
```

Session Example (cont.)


```
<?php
  session_start();
?>
<html>
<head><title>Testing Sessions page 2</title></head>
<body>
<?php
  echo "session_var = {$_SESSION['session_var']}<br>\n";
  echo "form_var = {$_POST['form_var']}<br>\n";
?>
</body></html>
```

(test me here)

Outline

Connecting to the Database

- Inline error handling

```
$conn_str = "host=$host port=$port user=$user
             password=$password";
$connection = pg_connect($conn_str)
                or die("Couldn't connect to server");
```

- Testing for errors

```
if (!$connection = pg_connect($conn_str))
{
   $message = pg_last_error();
   echo "$message<br>";
   die();
}
```

Connecting to the Database (cont.)

- Selecting the database

```
$dbname = "bank";
$conn_str = "host=$host port=$port user=$user
             password=$password dbname=$dbname";
$connection = pg_connect($conn_str);
```

- Closing the connection

```
pg_close($connection);
```

Sending Queries


```
$result = pg_query($query)
              or die("Couldn't execute query.");
```

or

```
$result = pg_query($connection,$query)
              or die("Couldn't execute query.");
```

- For queries that return data, `$result` is a pointer to the results
- For queries that do not return data, `$result` contains true is the statement succeeded, false if otherwise

Fetching the Results

- Results are fetched into an array

```
$row = pg_fetch_assoc($result);
while($row)
{
    $valor = $row["atributo"];
    echo "<p>$valor</p>";
    $row = pg_fetch_assoc($result);
}
```

Example


```
<table border=1>
<tr><td><b>Name</b></td><td><b>Street</b></td>
<td><b>City</b></td></tr>
<?php
   $sql="select * from customer";
   $result = pg_query($sql);
   echo "N. of results: " . pg_num_rows($result)";
   echo "N. of columns: " . pg_num_fields($result)";
   while($row_array = pg_fetch_assoc($result))
   {
     echo "<tr>";
     echo "<td>{$row_array['customer_name']}</td>";
     echo "<td>{$row_array['customer_street']}</td>";
     echo "<td>{$row_array['customer_city']}</td>";
     echo "</tr>";
   }
?>
</table>
```

Processing Forms

- An HTML form

```
<form action="processform.php" method="post">
<p>Your name: <input type="text" name="name" /></p>
<p>Your age: <input type="text" name="age" /></p>
<p><input type="submit" /></p>
</form>
```

- Program *processform.php* will have access to the forms fields through the arrays
  - `$_POST`: contains elements for all the fields contained in a form if the form uses method POST
  - `$_GET`: contains elements for all the fields contained in a form if the form uses method GET
  - `$_REQUEST`: contains all elements in `$_POST`, `$_GET` and `$_COOKIE`
- Example: `$_POST['name']`

Processing Forms Example


```
<?php

echo "<html>
      <head><title>Customer Address</title></head>
      <body>";

foreach ($_POST as $field => $value)
{
  echo "$field = $value<br>";
}
?>
</body></html>
```

(test me here)

Using Checkboxes

- Checkboxes allow multiple choice values

```
<input type='checkbox' name='type[account]'
    value='account'>
<input type='checkbox' name='type[loan]'
    value='loan'>
```

- In this case, the variable `$_POST['type']` is also an array, such that

```
$_POST[type][account] == 'account'
$_POST[type][loan] == 'loan'
```

if both "account" and "loan" where selected.

Cleaning the Input Data

- PHP provides some useful functions to clean the user input
- strip_tags: removes all HTML tags
- htmlspecialchars: converts all special characters to safe HTML entities
- Example:

```
$last_name = strip_tags($last_name,"<b><i>");
$last_name = htmlspecialchars($last_name);
```

References

- The W3Schools PHP Tutorial
  - `http://www.w3schools.com/php/default.asp`
- The PHP Web site
  - `http://www.php.net/`

The End