Programação Web Servidor

Desenvolvimento Web e Multimédia, 1º ano – 2º semestre

PHP

The PHP Hypertext Processor

POLITÉCNICO DE LEIRIA | ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Copyright

Author(s):

- **Alexandrino Gonçalves**
- **Vítor Carreira**

Contributor(s):

- **Ana Nogueira**
- **João Real**
- **Michael Pinheiro**
- **Norberto Henriques**
- **Telmo Marques**

Revised on: 14/04/2023

Dynamic web page

- A dynamic web page is a kind of web page where information is prepared (fetched/created/aggregated) in real time according to information available at the moment, context, user preferences or a combination of all.

Server-side languages

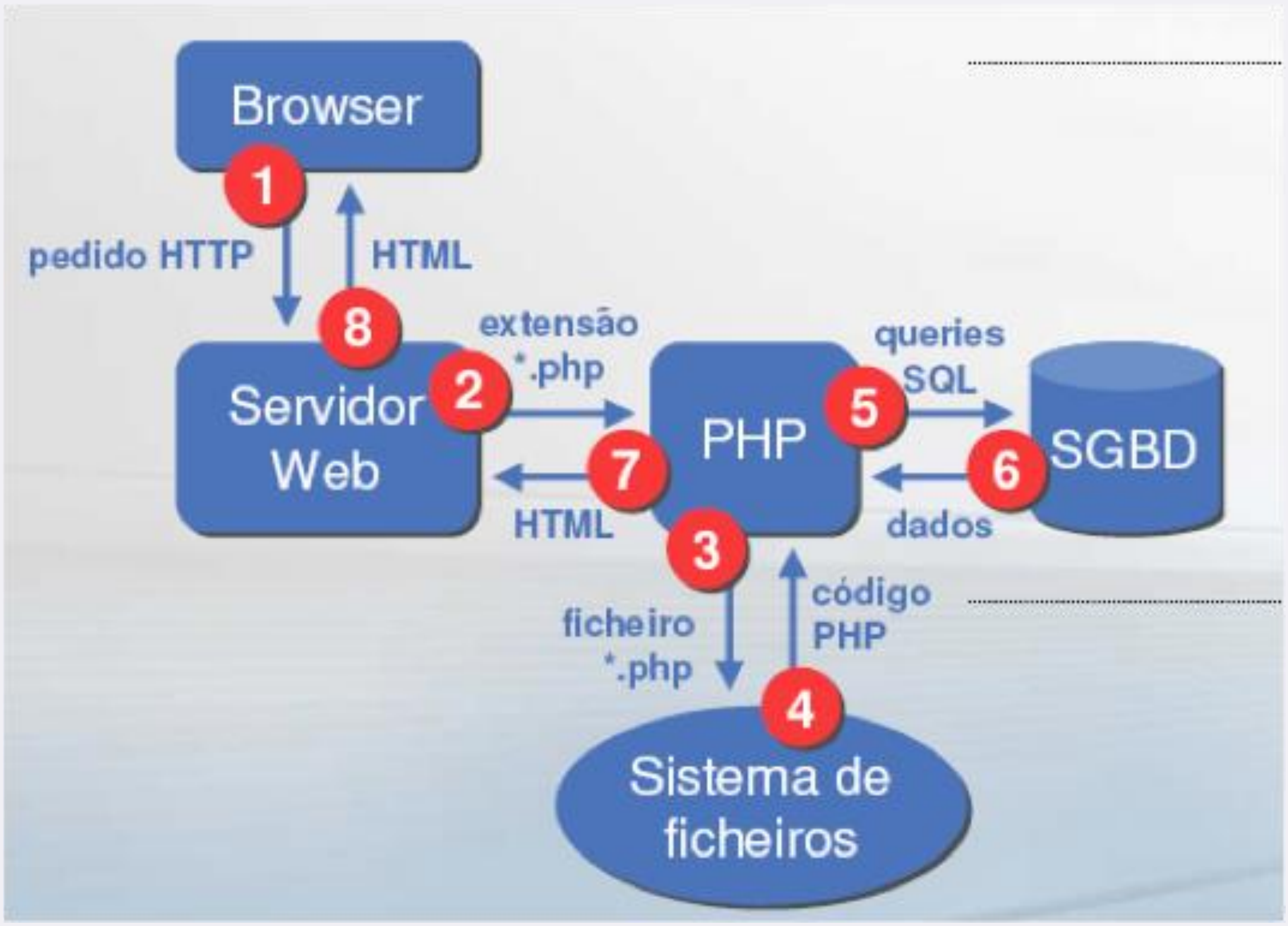
- Languages/frameworks:
  - PHP
  - JavaScript (frameworks: Node.js)
  - Python
  - Go/Golang
  - Java
  - C#
  - Etc.

PHP

PHP – PHP Hypertext Preprocessor

- Server-side scripting language (can also be used for desktop applications)
- Supports both procedural and object-oriented paradigms
- Specially designed for dynamic web page creation
- Cross operating systems: windows, linux, macOS
- Supported on a diversity of web servers: apache, IIS, etc.
- Support for multiple Database Management Systems (DBMS): mySQL, Oracle, SQLServer, etc.

PHP: Architecture



PHP: Syntax

- PHP script starts with *<?php* and ends with *?>*

```
<?php
  // PHP code
?>
```

- Statements must end with “;”

- PHP keywords classes, functions, and user-defined functions <u>are not case-sensitive</u>; but all variable names <u>are case-sensitive</u>

- The file extension is ".php“ (default)

- Now, each file must be triggered by: **http://server/file.php**

PHP: Syntax

- PHP tag and comments

```
<!DOCTYPE html>
<html>
   <head>
      <title>PHP Teste</title>
   </head>
   <body>
         <?php
            //Isto é um comentário
            # Isto é outro comentário
            /* Isto é um comentário em bloco */
            echo '<h3>Desenvolvimento Web Multimédia</h3>';
            echo '<p>Programação Web Servidor</p>';
         ?>
   </body>
</html>
```

PHP: Output

- echo & print

```
<!DOCTYPE html>
<html>
   <head>
      <title>PHP Output</title>
   </head>
   <body>
     <?php
        echo "<h2>Programação Web Servidor</h2>";
        echo "Output para o cliente!<br>";
        echo "Output ", "com ", "vários ", "parametros.";
        print "<br>Também funciona com o print.";
     ?>
   </body>
</html>
```

PHP: Variables 1/3

- No need to declare variables (just assign a value)
- Weakly typed
- Variable’s naming rules:
  - MUST start with $
  - Can include letters, numbers and _
  - Cannot start with digits
  - Case sensitive
- Use unset(<varname>) to explicit destroy a variable (e.g. unset($age);)
- Use *var_dump($age);* to know the data type of a variable

```
$age = 12;
$price = 2.55;
$number = -2;
$var = "Jones";
$logic = true;
$var = 5;
```

PHP: Variables 2/3

```
$a="PWS";
echo "$a is cool<br>";
echo $a." is cool";
// They have the same output
```

PHP: Variables 3/3

```
$a="hello";
$$a="world";
echo "$a ${$a} <br>";
echo "$a $hello <br>";
// They have the same output
```

PHP: Constants

- Constants are specified using keyword **define**

```
define("UC","Programação Web Servidor");

define("AGE",22);

...

echo UC;

echo AGE;
```

PHP: Strings 1/2

- Strings are a sequence of characters enclosed by " or ‘
- **String concatenation operator: .**
- String functions: *strlen*, *strpos*, *implode*, *etc*.

```
$string = 'Hello World!';
$string = 'It is Tom\'s house';
$string1 = 'Hello';
$string2 = 'World!';
$stringall = $string1.' '.$string2;
echo strlen("Hello world!"); // 12
echo strpos("Hello world!","world"); // 6
echo str_word_count("Hello world!"); // 2
```

PHP: Strings 2/2

- **Strings enclosed by " are interpreted**
- **Strings enclosed by ‘ are not interpreted**

```
$name = "Manel";

echo 'O teu nome é $name';
// Outputs "O teu nome é $name"

echo "O teu nome é $name";
// Outputs "O teu nome é Manel"
```

PHP: Data types 1/5

- Like JavaScript, PHP is weakly typed
- The data type is inferred by the value assigned to the variable
- Internal data types: *String, Integer, Float, Boolean, Array, Object*
- Special data types:
  - *NULL* (no value assigned)
  - *NaN* (not a number)
  - *Resource* (represents a handler to external resources like opened files, database connections, etc.)

PHP: Data types 2/5

- Data type conversion and test functions:
  - string **gettype**(mixed var) - Get var’s data type;
  - bool **settype**(mixed var, string type) - Change var’s data type;
  - **is_array**() - Checks whether the variable is an array;
  - **is_double**(), **is_float**(), **is_real**() - Checks whether the variable is a float;
  - **is_long**(), **is_int**(), **is_integer**() - Checks whether the variable is an integer;
  - **is_string**() - Checks whether the variable is a string;

PHP: Data types 3/5

- Data type conversion and test functions:
  - **is_bool**() - Checks whether the variable is a boolean;
  - **is_object**() - Checks whether the variable is an object;
  - **is_resource**() - Checks whether the variable is a resource;
  - **is_null**() - Checks whether the variable is null;
  - **is_scalar**() - Checks whether the variable is a scalar, that is, an integer, boolean, string, or float;
  - **is_numeric**() - Checks whether the variable is any kind of number or a numeric string;
  - **is_callable**() - Checks whether the variable is the name of a valid function;

PHP: Data types 4/5

- Test/change variable state:
  - bool **isset**(mixed var) – returns true if the variable var is defined;
  - void **unset**(mixed var) – destroys the variable var;
  - bool **empty**(mixed var) – returns true if the variable var does not exist or is not initialized;
- Conversion functions:
  - int **intval**(mixed var[, int base]) - converts var to an int value;
  - float **floatval**(mixed var) - converts var to a float value;
  - string **strval**(mixed var) - converts var to a string value;

PHP: Data types 5/5

- Sometimes we need to cast a value into another data type

```
// Cast a float to int
$x = 10.5;
$int_cast = (int)$x;
echo $int_cast."<br>"; // Outputs 10

// Cast a string to int
$str = "12";
$int_cast = (int)$str;
echo $int_cast+10; // Outputs 22
```

PHP: Operators

- <u>Arithmetic operators</u>: +, -, *, /, %, ** (exponentiation)
- <u>Comparison operators</u>: ==, >, <, >=, <=, != (or <>), ===, !==, <=>

```
echo (5 <=> 10); // -1 since 5<10
echo (5 <=> 5); // 0 since 5<10
echo (10 <=> 5); // 1 since 10>5
```

- <u>Logical operators</u>: && (and), || (or), xor, !

PHP: Date/Time

- Current time and date

```
date(format)
```

```
// For the day: 28/03/2023
echo date("l")."<br>"; // Tuesday
echo date("d")."<br>"; // 28
echo date("m")."<br>"; // 03
echo date("y")."<br>"; // 23
echo date("Y")."<br>"; // 2023
echo date("d\/m\/Y\, H:i:s"); // 28/03/2023, 10:05:14
// H-Hour from 0 to 23
```

PHP: Control structures

- *if* statement

```
if ($country == "Germany" )
{
      $message = "Willkommen!";
} elseif ($country == "France" )
  {
      $message = "Bienvenue!";
  } else
     {
       $message = "Welcome!";
     }
echo "$message<br>";
```

PHP: Control structures

- *if* statement

```
$a = "12";
$b = 10;
if ($a == $b+2)
      echo "DWM";
else
      echo "PWS";
```

Outputs **DWM**

```
$a = "12";
$b = 10;
if ($a === $b+2)
      echo "DWM";
else
      echo "PWS";
```

Outputs **PWS**

PHP: Control structures

- *switch* statement

```
switch($country)
{
   case "Germany": $salestaxrate = 0.16;
                   break;
   case "Portugal":$salestaxrate = 0.23;
                   break;
   default: $salestaxrate = 0.19;
            break;
}
$salestax = $orderTotalCost * $salestaxrate;
```

PHP: Control structures

- *while* statement

```
$i = 1;
echo "<table border='1'>";
while($i <= 10)
{
      $y=$i**2;
      echo "<tr><td>$i</td><td>$y</td></tr>";
      $i++;
}
echo "</table>";
```

PHP: Control structures

- *do...while* statement

```
$i = 1;
echo "<table border='1'>";
do
{
      $y=$i**2;
      echo "<tr><td>$i</td><td>$y</td></tr>";
      $i++;
} while($i <= 10);
echo "</table>";
```

PHP: Control structures

- *for* statement

```
for ($i = 1; $i <= 10; $i++)
{
      $y=$i**2;
      echo "The square of $i is $y <br>";
}
for ($i = 0, $j = 1; $t <= 4; $i++, $j++)
{
      $t = $i + $j;
      echo "$t<br>";
}
```

PHP: Control structures

- *break* and *continue* statements

```
$max = 10;
for ($i = 1; $i <= $max; $i++)
{
      if ($i == 4) {
            continue;
         // break;
      }
      $y=$i**2;
      echo "The square of $i is $y <br>";
}
```

PHP: Functions 1/6

- Syntax: `function <name>(args) { //code }`

```
function finalCost($value, $tax)
{
  $total = $value * (1+$tax);
  return $total;
}

$tennis=100;
$iva = 0.23;
echo finalCost($tennis,$iva);
```

PHP: Functions 2/6

- Function default values

```
function add_2_numbers($num1 = 1, $num2 = 1)
{
  $total = $num1 + $num2;
  return $total;
}
echo add_2_numbers().'<br>'; // Outputs:2
// or
echo add_2_numbers(3).'<br>'; // Outputs:4
// or
echo add_2_numbers(4,2).'<br>'; // Outputs:6
```

PHP: Functions 3/6

- Local vs Global variables

```
<?php
$VAT=0.23; // Global variable

function cost_with_vat($cost){
      global $VAT; // references global variable
      $total_with_vat = $cost + $cost * $VAT;
      return $total_with_vat;
}
$total=cost_with_vat(199.99);
?>
```

PHP: Functions 4/6

- *static* call can be useful in recursive functions

```
<?php
function staticVars() {
      static $x=0;
      $y=0;
      $x++;$y++;
      echo "<br>$x $y<br>";
}
staticVars(); // Outputs: 1 1
staticVars(); // Outputs: 2 1
?>
```

PHP: Functions 5/6

- Arguments

```
<?php
function increment($num, $amount = 1)
{
  $num = $num + $amount;
}
$num = 10;
echo $num.'<br>'; // Outputs 10
increment ($num, 5);
echo $num.'<br>'; // Outputs 10
?>
```

PHP: Functions 6/6

- Pass by reference

```
<?php
function increment(&$num, $amount = 1)
{
   $num = $num + $amount;
}
$num = 10;
echo $num.'<br>'; // Outputs 10
increment ($num, 5);
echo $num.'<br>'; // Outputs 15
?>
// In PHP arrays are not passed by reference
```

PHP: Arrays1/8

- Simple arrays

```
$animals = array("cat","tiger","elephant");
// or
$animals[0] = "cat";
$animals[1] = "tiger";
$animals[2] = "elephant";
--------------------------------
for($i=0; $i<count($animals); $i++)
       echo $animals[$i]." ";
// or
foreach ($animals as $animal)
   echo $animal.' ';
```

```
$animals = array("cat","tiger","elephant");
$animals[20] = "dog";
count($animals); // Returns 4
end($animals);
key($animals); // Returns 20
```

PHP: Arrays 2/8

- Associative arrays (usually keys are strings)

```php
$airlines = array("BA" => "British Airways",
"LH" => "Lufthansa", "AF" => "Air France");
// or
$airlines['BA'] = "British Airways";
$airlines['LH'] = "Lufthansa";
$airlines['AF'] = "Air France";
```

PHP: Arrays 3/8

- Sort operations on simple arrays

```
sort($animals);
rsort($animals); // Reverse sort
```

- Sort operations on associative arrays:
    - By value: `asort`, `arsort` (reverse order)
    - By key: `ksort`, `krsort` (reverse order)

PHP: Arrays 4/8

- Foreach and associative arrays

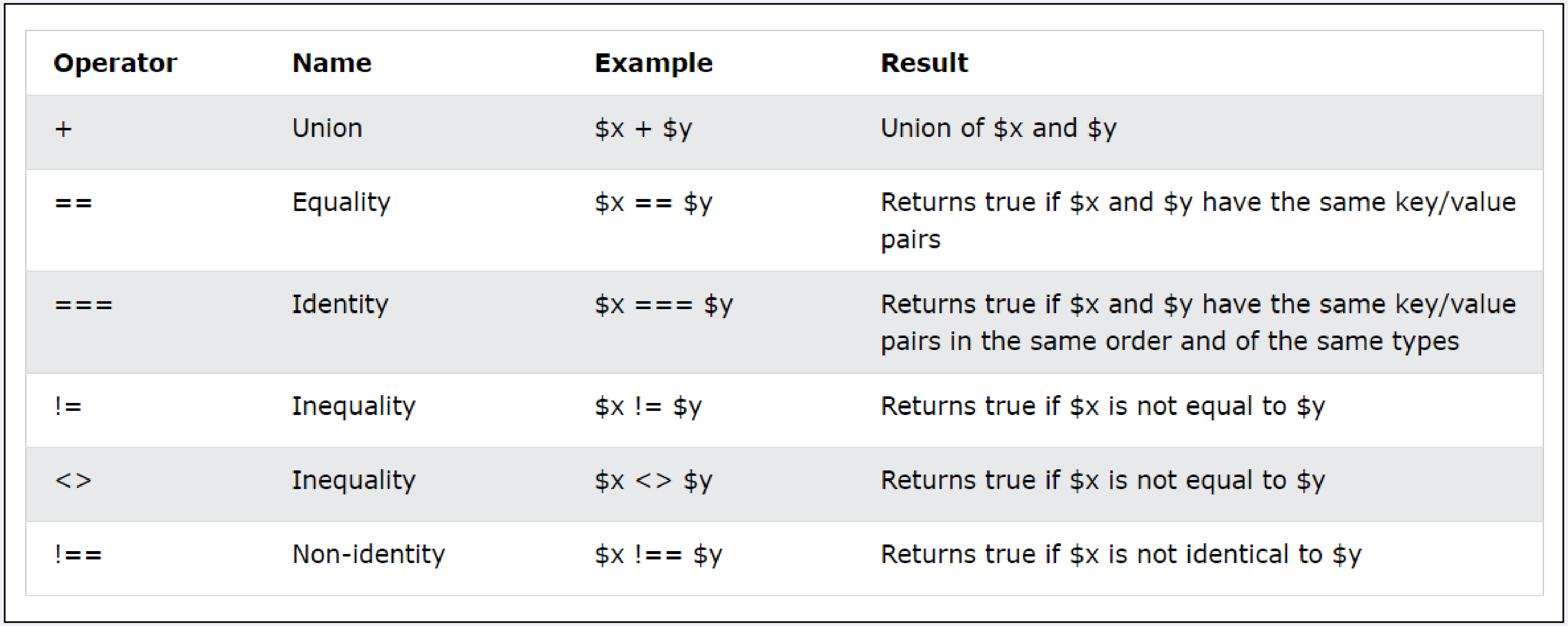
```
$airlines = array("BA" => "British Airways","LH" =>
"Lufthansa", "AF" => "Air France");
krsort ($airlines);
foreach($airlines as $symbol => $name)
{
  echo "$name ($symbol)<br>";
}
```

PHP: Arrays 5/8

- Arrays support iterators: `reset(), current(), prev(), next(), end(), sizeof()` (same as `count()`)

```
reset($airlines); // moves to the first element
$value = current($airlines);
echo "$value<br>"; // current array element
$value = next($airlines);
echo "$value<br>";
$value = next($airlines);
echo "$value<br>";
```

PHP: Arrays 6/8



Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	Returns true if $x is not equal to $y
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y

PHP: Arrays 7/8

- Multi-dimensional arrays

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

PHP: Arrays 8/8

- Multi-dimensional arrays

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

PHP header

- Refresh - allows to refresh a page after some delay

```
<?php
  header('Refresh: 10; url=http://www.example.com/newpage.php');
?>
<!DOCTYPE html>
<html>
  <head><meta ...><title>....</title></head>
  <body>
    <p>You will be redirected in 10 seconds</p>
  </body>
</html>
```

- HTML equivalent refresh header

```
<meta http-equiv="refresh" content="10; url=http://www.example.com/newpage.php">
```

PHP: Code reuse 1/2

- PHP promotes code reuse:
  - **include**(<file>): triggers a warning if the file doesn’t exist;
  - **require**(<file>): throws an error if the file doesn’t exist;
  - **require_once**(<file>): identical to *require()* except PHP will check if the file has already been included, and if so, not include (require) it again. Recommended for bootstrapping code;
  - **include_once**(<file>): identical to the previous.

PHP: Code reuse 2/2

```
<?php
require_once 'header.php' ;
?>
<!-- page content -->
<p>Welcome to the home of PWS.</p>
<?php
require_once 'footer.php';
?>
```

HTML Form 1/2

- Allow users to send data to a web server
- Tag: **<form>**
- Main attributes:
  - **action** - URL where the form-data is sent to
  - **method** - type of request: GET (by default) or POST
  - **enctype** - specifies how form-data should be encoded before sending it to the server

HTML Form 2/2

- **enctype** attribute allowed values:
  - **application/x-www-form-urlencoded** - All characters are encoded before sent (by default)
  - **multipart/form-data** - No characters are encoded. <u>This value is required when you are using forms that have a file upload control</u>
  - **text/plain** - Spaces are converted to "+" symbols, but no special characters are encoded

PHP and Forms

- PHP provides 5 built-in **superglobal** variables for Form processing:
  - **$_POST** - an associative array of variables passed to the current script via the HTTP POST method
  - **$_GET** - an associative array of variables passed to the current script via the URL parameters (HTTP GET request)
  - **$_COOKIE** - an associative array of variables passed to the current script via HTTP Cookies
  - **$_REQUEST** - an associative array that by default contains the contents of $_GET, $_POST, $_COOKIE
  - **$_FILES** - An associative array of files uploaded to the current script via the HTTP POST method
- The key used to fetch the value is the name of the control (attribute **name**). Example: `$_POST["email"]`

POST method 1/3

- File "form_post.html"

```
<form action="process_form_post.php" method="post">
<div>
       <label for="firstName">First Name:</label>
       <input type="text" name="firstName" id="firstName">
</div>
<div>
       <label for="age">Age:</label>
       <input type="text" name="age" id="age">
</div>
<div>
       <label for="pass">Password:</label>
       <input type="password" name="pass" id="pass">
</div>
       <input type="submit">
</form>
```

POST method 2/3

- File "process_form_post.php"

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>PHP Forms</title>
</head>
<body>
<h1>Forms in PHP</h1>
<?php
echo "<p>Welcome ".$_POST["firstName"].".</p>";
echo "<p>You have ".$_POST["age"]." years old.</p>";
?>
</body>
</html>
```

POST method 3/3

- When the user submits the form, none of the fields will be part of the URL. E.g.:

  http://localhost/process_form_post.php

- There is no limit (client side) for the size of the request
- The content of a request (POST) is normally limited by the server on a byte size basis in order to prevent a type of DoS attack.

GET method 1/3

- File "form_get.html"

```
<form action="process_form_get.php" method="get">
<div>
       <label for="firstName">First Name:</label>
       <input type="text" name="firstName" id="firstName">
</div>
<div>
       <label for="age">Age:</label>
       <input type="text" name="age" id="age">
</div>
<div>
       <label for="pass">Password:</label>
       <input type="password" name="pass" id="pass">
</div>
       <input type="submit">
</form>
```

GET method 2/3

- File "process_form_get.php"

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>PHP Forms</title>
</head>
<body>
<h1>Forms in PHP</h1>
<?php
echo "<p>welcome ".$_GET["firstName"].".</p>";
echo "<p>You have ".$_GET["age"]." years old.</p>";
?>
</body>
</html>
```

GET method 3/3

- When the user submits the form, every field will be part of the URL. E.g.:

  http://localhost/process_form_get.php?firstName=ana&age=22

- **This is not a good option for sending sensitive data** (passwords, uids, etc).

- Although the specification of the HTTP protocol does not specify any maximum length, practical limits may be imposed by web browsers and server software (about 2000 characters).

Uploading files 1/2

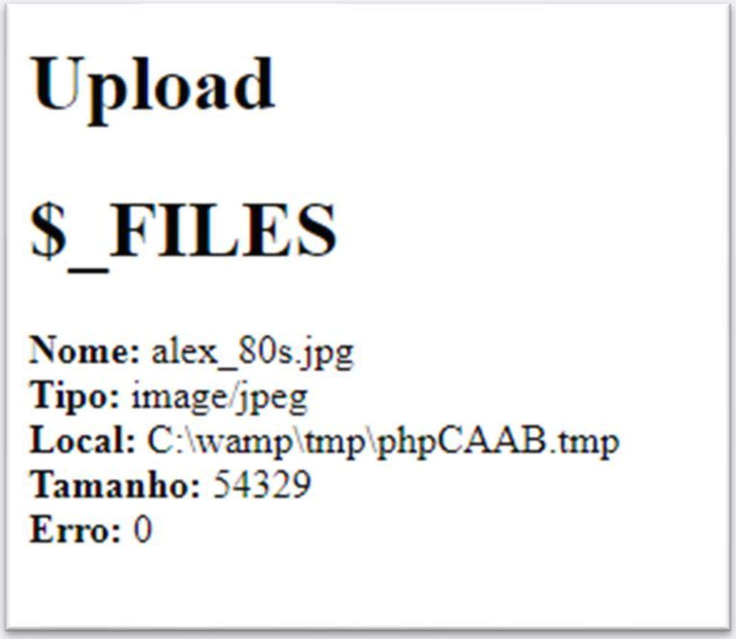
- File "form_file.html"

```
<form action="upload.php" enctype="multipart/form-data"
method="post" >
<div>
  <label for="description">Image description:</label>
  <input type="text" id="description" name="description">
</div>
<div>
  <label for="image">Image:</label>
  <input type="file" name="image" id="image">
</div>
<div>
  <input type="submit" value="Send Image">
</div>
</form>
```

Uploading files 2/2

- File "upload.php"

```
<?php
echo '<h1>$_FILES</h1>';
echo "Nome: ".$_FILES["image"]["name"];
echo "<br>Tipo: ".$_FILES["image"]["type"];
echo "<br>Local: ".$_FILES["image"]["tmp_name"];
echo "<br>Tamanho: ".$_FILES["image"]["size"];
echo "<br>Erro: ".$_FILES["image"]["error"];
?>
```



Upload

$_FILES

Nome: alex_80s.jpg
Tipo: image/jpeg
Local: C:\wamp\tmp\phpCAAB.tmp
Tamanho: 54329
Erro: 0

State and HTTP 1/2

- HTTP is a stateless protocol
- There is no built-in way of maintaining state between two transactions
- There is no automatic link or association between subsequent requests from the same user

State and HTTP 2/2

- How to implement a Shopping Cart feature that needs to keep state across requests?
  - Client-side: cookies
  - Server-side: sessions

Cookies 1/2

- Cookie is a kind of variable (name-value pair) sent by the server on each request and is stored on the client’s web browser.
- When the browser fetches a web page, it sends along with the request all cookies stored for the page’s domain/path
- Cookies have attributes for:
  - **domain and path** - defines the cookie scope;
  - **expiration date** - tells the browser when to delete the cookie;
  - **security** - restricts the cookie’s usage (secure connections only, http protocol only).

Cookies 2/2

- Limitations:
  - 4KB of maximum storage for each cookie (for maximum compatibility).
  - Browser's:
    - There is a limit per domain. Usually, a single domain can store at least 20 cookies;
    - A global cookie limit which erases the oldest cookies when the limit is reached.
  - Users can delete or disable cookies.

PHP Cookies 1/2

- Function `setcookie()` - sends a cookie to the client (`setcookie` <=> `header(“Set-Cookie: ...”)`).

- The superglobal associative array `$_COOKIE` keeps track of the cookies sent by the client.

- The `setcookie()` function <u>must appear before the <html> tag</u>

PHP Cookies 2/2

```
<?php
  $counter = 0;
  if (isset($_COOKIE['counter']))
       $counter =  $_COOKIE['counter'];
  $counter++;
  // set a cookie called counter. Cookie expires after 300s
  setcookie('counter', $counter, time() + 300);
?>
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>PHP: Cookies</title>
  </head>
<body>
  <h1>Welcome. This is your visit #<?php echo $counter ?></h1>
</body>
</html>
```

Sessions

- Session control allows a web server to track a user during a single session on a website;
- The session ID is generated by the server and stored on the client side for the lifetime of a session. It can be either stored on a user’s computer in a cookie or passed along through URLs;
- The session ID acts as a key to register variables called session variables;
- The session variables are stored at the server;
- A sessions has an implicit timeout, after which it is destroyed;
- The session ID is the only information visible at the client side.

PHP Sessions

- Sessions in PHP are represented by a unique session ID (cryptographically 32-digit hexadecimal random number).
- The basic steps of using sessions are:
    1. Starting a session;
    2. Registering session variables;
    3. Using session variables;
    4. Deregistering variables and destroying the session.
- <u>Note</u>: these steps don't necessarily occur in the same script.

Starting a session 1/2

- **`session_start()`** - creates a session or resumes the current one based on a session identifier (PHPSESSID) passed via a cookie or passed via a GET or POST request.

- When a session is resumed, the **`$_SESSION`** super global associative array is filled with the session variables associated to the session id.

- **session_start()** must be called before outputting anything to the browser.

Registering session variables

- Session variables are stored in the associative array $_SESSION

```
<?php
  session_start();
  ...
  $_SESSION['Username'] = $username;
  $_SESSION['LastOperation'] = time();
  $_SESSION['ShoppingCart'] = $new_product;
  ...
?>
...
```

Using session variables

- Always check if session variables have been set (`isset()` or `empty()`)

```
<?php
 session_start(); // Don’t forget to start/resume session
?>
...
<?php
  if (isset($_SESSION['authenticated'])) {
    printf('<p>Welcome %s, <a href="logout.php">Logout</a></p>',
      $_SESSION['username']);
}
?>
...
```

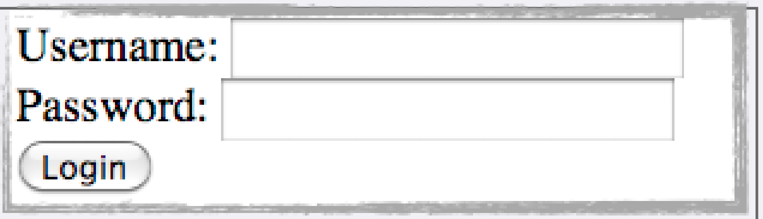
Deregistering variables and destroying the session

- Destroy each session variable individually by calling **unset($_SESSION["<var_name>"])** or use the following shortcut to destroy all session variables attached to a session: **`$_SESSION = array()`**;
- At the end invalidate the session id by calling **`session_destroy()`**.
- **NEVER** call **`unset($_SESSION)`** - this will disable further sessions.

```
<?php
  session_start();
  $_SESSION = array();
  session_destroy();
?>
```

Example (login.php) 1/4

```
<?php
session_start();
if (isset($_SESSION['authenticated'])) {
        header('Location: private.php');
        exit(0); }?>
<html>
<head><title>...</title></head>
<body>
<form action="auth.php" method="post">
<div><label for="username">Username:</label>
<input type="text" name="username" id="username"></div>
<div><label for="pass">Password:</label><input type="password"
name="pass" id="pass"></div>
<div><input type="submit" value="Login"></div>
</form>
<?php
  if (isset($_SESSION['errors'])) {
  echo "<div>Errors:";
  foreach ($_SESSION['errors'] as $field => $error)
        echo "<p>$field: $error</p>";
        echo "</div>";
}?></body></html>
```



Example (auth.php) 2/4

```
<?php
  session_start();
  $_SESSION['errors'] = array(); // Cleanup previous errors
  if  (isset($_POST['username'])) $username = trim($_POST['username']);
    else $username = "";
  if  (isset($_POST['pass'])) $password = trim($_POST['pass']);
    else $password = "";
if (strlen($username) == 0)
     $_SESSION['errors']['username'] = 'Empty username';
  if (strlen($password) == 0)
     $_SESSION['errors']['pass'] = 'Empty password';
  if (count($_SESSION['errors']) == 0) {
     if (strcmp($username, $password) == 0) {// Some dummy authentication
        $_SESSION['authenticated'] = true;
        $_SESSION['username'] = $username;
        header('Location: private.php');
      }
        else
          $_SESSION['errors']['auth'] = 'Authentication failed';
  }
  if (count($_SESSION['errors']) != 0) {
     header('Location: login.php');
     exit(0); }?>
```

Example (private.php) 3/4

```
<?php
  session_start();
  if (!isset($_SESSION['authenticated'])) {
       header('Location: login.php');
       exit(0);}
?>
<!DOCTYPE html>
<html>
  <head><meta ...><title>....</title></head>
  <body>
<?php
  printf('<p>Welcome %s, <a href="logout.php">Logout</a></p>',
       $_SESSION['username']);
  echo "<p>Session id:".session_id()."</p>";
?>
</body>
</html>
```

Example (logout.php) 4/4

```
<?php
  session_start();
  $_SESSION = array();
  session_destroy();
  header('Location: login.php');
?>
```

References

- Robin Nixon, "Learning PHP; MySQL and JavaScript: With jQuery; CSS and HTML5", 5th. Edition. O'Reilly, 2018

- PHP Documentation
  - http://www.php.net/
  - Function reference: http://www.php.net/manual/en/funcref.php