PHP Introduction

DWES UD2

1.- Introduction

Static web pages, with a **.html** extension, can be executed without having a web server.

It is the browser that interprets the code of these files.

We have seen this in the first exercise in which HTML tags were tested.

Double-clicking a .html file opens the file in the browser directly.

1.- Introduction

In order to execute .**php** files, the web server must execute these files.

From now on our projects must be saved in the web page repository.

In **Apache** the web page repository is the **htdocs** directory.

For now we will work assuming that on our web server we only have one web application, **the project we are working on at all time.**

1.- Introduction

We have to make sure that there are no files in the htdocs directory that are not from our project.

When you see **virtual hosts** in the **DAW** module (web application deployment) we can have different projects at the same time on the same web server.

Exercise: Go into the htdocs directory, cut out all the elements there are and paste them into a folder called htdocs_original in the main xampp directory.

(We save these files in case at any time we touch something we shouldn't, we can recover them to enter the information pages).

2.- Integration of PHP into HTML

In this course we are going to use **PHP** as a programming language for dynamic web pages.

As mentioned in the previous topic, the files to create dynamic web pages with PHP will have the .**php** extension but inside they will contain **HTML** code combined with **PHP** code.

The first thing we are going to see is how we can **integrate** PHP code within HTML code, but we will also see how to integrate HTML code within PHP

2.- Integration of PHP into HTML

Exercise: Create a file in the **htdocs** directory called **prueba.php**

```
<!doctype html>
<html>
        <head>
                <meta charset="UTF-8">
                <title>Primera prueba php</title>
        </head>

        <body>
                Este es un archivo php que se encuentra en el servidor.
    </body>
</html>
```

Openr the browser and access to the URL: **http://localhost/prueba.php**

2.- Integration of PHP into HTML

It is possible and quite common that there is only PHP code in a .php file.

This happens when you are just trying to process something.

For example when calling a function.

Ejercicio: Create a file called info.php placed inside the htdocs directory with the following code:

```
<?php
        phpinfo();
?>
```

Open the browser and access the URL: **http://localhost/info.php**

2.- Integration of PHP into HTML

The best place to consult the syntax and functions of PHP is in the official documentation, which is always updated and most of it in Spanish.

You can find other websites with documentation but it is possible that it is for older versions of PHP.

The good thing about the official site is that it tells you the versions in which each of the functions works and whether its use is discouraged or not.

2.- Integration of PHP into HTML

PHP inside of HTML

```
<article>
<?php

        //code with PHP instructions

?>
</article>
```

2.- Integration of PHP into HTML

HTML inside of PHP

```
<?php

        echo ‘<h1>Bienvenido a mi página web</h1>’;

?>
```

3.- Basic PHP Features

Comments

One line comments:

```
// Función para loguearse
```

Multi-line comments:

```
/* Script desarrollado por:
    Pau Medina
    fecha: 10-9-23 */
```

The commented code in PHP will not appear in the final HTML document that is generated.

*Remember that HTML comments will appear: <!-- -->

3.- Basic PHP Features

Variables and data types

Variables in PHP are case sensitive and always begin with the **$** symbol

After the **$** there must be a **letter** or the symbol _

The rest of the characters can only be letters, numbers and the symbol _

$edad	$nombreCalle
$_piso	$i
$variable2	$primer_apellido

The following variables would not be valid:

$cantidad pelotas	$3tipos
$valor?	$direccion.usuario

3.- Basic PHP Features

Variables and data types

PHP is a weakly typed programming language.

The type of data to be stored in each variable is not specified.

In the variables you can change the type of data that is contained according to interest.

To assign value to variables, use the = operator.

```
$variable = 5;

$variable = “Federico”;

$variable = TRUE;
```

3.- Basic PHP Features

Variables and data types

The types of data that can be stored in variables in PHP are:

- boolean: **TRUE/FALSE**, **0** = false, **any_other_number** = true.
- integer: number without decimals.
- float: number with decimals.
- string: quote delimited character set.
- null: special type that indicates that the variable has no value.

3.- Basic PHP Features

Variables and data types

```
$booleano = FALSE;

$edad = 58;

$kilos = 5.3;

$nombreCompleto = “Ana Gómez Parra”;

$otro = null;
```

3.- Basic PHP Features

Variables and data types

If an operation involves operations of different types, the PHP module will try to convert them to a common type (casting).

```
$cantidad = 3;
$precio = 1.6;

$total = $cantidad * $precio;
```

Exercise: In the **prueba.php** document, enter different instructions in to the body section of the page that operate with variables of different types. Access the URL http://localhost/prueba.php to see how the server reacts.

3.- Basic PHP Features

Variables and data types

Casting can be done forcefully

```
$cantidad = 3;
$precio = 1.6;

$total = $cantidad * (int)$precio;          //¿Cuánto valdrá $total?
```

In the documentation you can see the types of conversions allowed.

3.- Basic PHP Features

Expressions and operators

As in other programming languages, PHP also uses expressions, for example add, subtract, assign.

Operators will usually appear in expressions, which are similar to those of other programming languages.

- Assign: **=**
- Arithmetics: **+ - * / % ++ --**
- Comparation: **> < >= <= == === != !==** ==, !== same type and value
- Boolean Comparations: **&& || !**

In the documentation you can see the types of operators.

3.- Basic PHP Features

Scope of variables

Variables can be used at any point in the program; if they do not exist, they will be created the first time they are used, reserving space in memory.

Depending on the first place where the variable appears for the first time will decide where it can be used (visibility of the variable).

If the variable appears inside a function for the first time, that variable will be local to the function.

Once the function ends, that variable disappears and its value is lost.

3.- Basic PHP Features

Scope of variables

```
$a = 1;
$b = 2;

function prueba() {
        $c = $a;

        global $b;
        $c = $b;
}
```

3.- Basic PHP Features

Scope of variables

```
function contador() {
        static $a=0;

        $a++;
}
```

3.- Basic PHP Features

HTML code generation

We have already seen previously how to enter HTML code from PHP instructions.

We have used the **echo** statement. There are also instructions **print** and **printf**.

```
<?php
        $modulo = “DWES”;
        echo “<p> Módulo”;
        print $modulo;
        print “</p>”;
?>
```

Actually **echo** and **print** are not functions, so it is not necessary to put the ( );

3.- Basic PHP Features

HTML code generation

Through the operator . arguments can be concatenated/joined in **echo** and **print** statements.

```
<?php
        $modulo = “DWES”;
        echo “<p> Módulo” . $modulo . “</p>;
?>
```

3.- Basic PHP Features

HTML code generation

The **printf** function is much more complete than **echo** and **print**, but its use is usually limited to cases in which it is important to **format** the data to be displayed (number of decimal places, length...)

Documentation (format section).

```
<?php
        $ciclo = “DAW”;
        $modulo = “DWES”;
printf(“%s es un módulo de %d curso de %s”, $modulo, 2, $ciclo);
?>
```

3.- Basic PHP Features

Text Strings

To declare/use text strings you can use single quotes or double quotes. The difference is that if double quotes are used we can include variables inside and the PHP module will convert them into its value.

```
<?php
        $edad = 24
echo “Edad:  $edad”;
echo “<br>”;
echo “Juan tiene ${edad}años”;
?>
```

3.- Basic PHP Features

Programming tip: Text Strings

Given how the HTML language works and how it is coded, in which double quotes are very commonly used (argument-value in tags):

```
<a href=”fotos.php”><img src=”camara.png” alt=”Galería fotográfica”></a>
```

In PHP to declare and use strings it is recommended to use single quotes.

```
<?php
    echo ‘<a href=”fotos.php”><img src=”camara.png” alt=”Galería fotográfica”></a>’;
?>
```

If you want to print a single quote you will have to use the escape sequence: **\’**

3.- Basic PHP Features

Text Strings

Text strings can use two operators unique to them:

Concatenate: **.**

Concatenate and assign: **.=**

```
<?php
        $a = “Módulo”;
        $b = $a . “DWES”;
        $a .= “DWES”;
?>
```

¿What contains $a and $b?

3.- Basic PHP Features

Text Strings

For the use of strings there is a large set of functions, all of them can be consulted in the Official documentation, depending on how we are going to use them during the course, they will be explained..

Below are two examples:

```
<?php
        $nombre = “Antonio”;
                echo strlen($nombre);

        $nombreMayus = strtoupper($nombre);
?>
```

3.- Basic PHP Features

Functions for data types

Using functions you can check the type of data that a variable is storing at a given time. It can be done in two ways, querying the data type or asking if it is a specific data type:

```
string gettype ($variable)
```

A value among the following will be obtained in the string: array, boolean, double, integer, object, string, null, resource or unknown type.

```
boolean is_array($variable)
boolean is_numeric()
boolean is_bool()
boolean is_integer()
```

3.- Basic PHP Features

Check if a variable exists

Sometimes it is interesting to know if the variable exists or not, for example when you receive a form and want to check that all the variables have arrived. For this there is a function that returns a boolean value: **isset**

There is also a function that allows you to destroy variables: **unset**

```
<?php
        $a = 25;
                $existe = isset($a);
                unset($a);
                $existe = isset($a);
?>
```

¿What value have $existe in each case?

3.- Basic PHP Features

Constant

Constants are identifiers that store a value that cannot change during the execution of the program. The reserved word **define** is used.

```
<?php
        define (“PI”, 3.141592);
        define (“NOMBRE”, “Luisa”, true);

        $radio = 5;
        $superficie = PI * $radio * $radio;
?>
```

If **true** is added to the end of the **define** statement it means that the name will be **CI** (Case Insensitive) NAME and Name represent the same value. Constants do not have a dollar sign as a prefix. **Integer, float, string, boolean**, and **null** types are allowed.

3.- Basic PHP Features

Dates and times

There is no data type to work with dates and times.

Stored as an integer (UNIX date - seconds since 1/1/70 00:00:00)

It is important to set the time zone correctly in order to display the data well.

```
<?php
date_default_timezone_set('Europe/Madrid');
?>
```

3.- Basic PHP Features

Dates and times

The **time** function returns the current date and time in UNIX format

```
<?php
        $start = time();
        /*
                Muchas instrucciones php
        */
        echo “Esta página se ha generado en ”. time()-$start .” segundos”;
?>
```

If you want to show detailed information about the date, you need to use a function that allows you to obtain specific data such as day, month, time... it is the **date** function.

3.- Basic PHP Features

Dates and times

The **date** function can work directly with the current date and time or with a date that is indicated directly to it.

```
<?php
    //Las dos instrucciones almacenarán los mismo
    $hoy = date("l, d M Y");
    $hoy = date("l, d M Y", time());
?>
```

The rest of the date formatting options can be found in the documentation.
If you want to generate a specific date you can use the mktime function

```
//mktime (hora, minutos, segundos, mes, dia , año)
$examen = mktime (18,5,0,11,8,2023);
```

Review Exercise

Used to define constants **date_default_timezone_set**

Returns a string in format **define**

Indicates if a variable is defined and its value is not null **settype**

Sets the type of a variable **date**

Get a text string from a date/time **printf**

Indicates whether a variable is of type string (string) **getdate**

Gets an array with information about the current date and time **isset**

Set the time zone **is_string**

3.- Basic PHP Features

Superglobals

PHP has a set of variables that can be used in **any scope** without needing to use the **global** keyword. These variables are **arrays** that contain interesting information about the system.

$_SERVER

Contains information about the server

Thanks to the **print_r** function you can display the entire contents of an array. This instruction is very useful for debugging tasks. var_dump it can also be used

```
<?
        print_r($_SERVER);
?>
```

3.- Basic PHP Features

Superglobals

They contain the variables that a php script can receive through the methods that indicate the name of the superglobal variable.

$_GET
$_POST
$_COOKIE

Combine the data from the previous three into a single array.

$_REQUEST

3.- Basic PHP Features

Superglobals

Contains the information of the files that have been sent with the POST method.

$_FILES

It is used to consult the session variables. Its use will be explained later.

$_SESSION

3.- Basic PHP Features

Inclusion of external files

When a program is developed it is common for it to grow and it is difficult to find some code.

It is also more than likely that there are pieces of the code that are needed in different places, in our case on different web pages of the application.

In PHP there are instructions that allow you to add the entire content of another file at a given point in the program.

include
include_once

require
require_once

3.- Basic PHP Features

Inclusion of external files

All these **functions** search for the indicated file, if found, they add the content to the point where the function call is located.

```
include(“ruta_archivo_php”);
require(“ruta_archivo_php”);
include_once(“ruta_archivo_php”);
require_once(“ruta_archivo_php”);
```

The **include** functions, if they do not find the file, give a warning but continue. However, if the **require** instructions do not find the file, they stop execution with an error.

Versions with **_once** serve to ensure that the file is only added once since if it is added more than once at different points it can cause errors (for example if the added file contains a function).

3.- Basic PHP Features

Inclusion of external files

It is common that if a file is designed so that whenever it is used it is done by including it with **include** or **require** that it is given the **.inc.php** extension.

For example:

formulario.inc.php

Exercise

Create a file called **prueba.inc.php** enter the following code **only**:

```
<h1>Esto viene de otro archivo php</h1>
```

Go to **prueba.php** that you already have and at the end of the body section add the following lines:

```
<?php
        include(“archivo.php”);
        include_once(“otro.php”);
        require(“prueba.inc.php”);
        require_once(“inventado.php”);
?>
```

Mandatory deliverable practice

When you finished the HTML5 reminder you had to make a small web application.

Logically, all **web pages** of the same **website** or **web application** should have the same style and therefore it is common for them to have the same header.

First move the HTML code of your website header to a different file called **cabecera.inc.php**.

Then, using the instructions for **including external files**, make sure that this header is added to all the web pages of your web application.