Desarrollo web en entorno servidor

UT 1.1 – Arquitectura y herramientas web

8 – PHP – Modularizar código – Fichero php.ini

IES Clara del Rey – Madrid

PHP – Modularizar código

En PHP, como en cualquier otro lenguaje, es una buena práctica modularizar el código, dividiéndolo en múltiples archivos.

Se dispone de cuatro instrucciones básicas para incluir unos archivos dentro de otros:

- include
- include_once
- require
- require_once

PHP – include – require

La instrucción "include" inserta el contenido del archivo especificado en el lugar donde se encuentre la sentencia.

```
<?php
include 'funciones.php'; // Incluye el archivo de funciones
resultadoOperacion(); // Usa una función de funciones.php
?>
```

Si el archivo no se encuentra, se produce un warning, y el script continúa.

Puede incluirse dos veces el mismo archivo, pero esto puede llevar a redefinir funciones o clases, lo que sí podría provocar errores.

La instrucción require es idéntica a include, pero si no se encuentra el archivo, se detiene el script con un error.

PHP – include_once – require_once

Include_once es como include, pero el archivo se incluirá sólo una vez.

```
<?php
include_once 'funciones.php'; // Incluye el archivo de funciones
resultadoOperacion(); // Usa una función de funciones.php
include_once 'funciones.php'; // Esto no hace nada
resultadoOtraOperacion(); // Usa otra función de funciones.php
?>
```

Igual que include, si el archivo no se encuentra, se produce un warning, pero el script sigue ejecutándose. Sólo se incluye el archivo donde se haya hecho la primera llamada a include_once.

require_once es idéntica a include_once, pero si no se encuentra el archivo se lanza un error.

PHP – Control de flujo e includes

Cualquiera de las instrucciones que hemos visto se puede usar dentro de control de flujo, de forma que se cargue o no un include en función de cierta condición.

```
<?php
# Comprobar si el usuario está autenticado, usando sesión, p. ej.
$usuarioAutenticado = isUserAutenticated();

# En función de si está o no autenticado, mostrar un include u otro
if ($usuarioAutenticado) {
        include 'dashboard.php';
} else {
        include 'login.php';
}
?>
```

PHP – Control de errores al usar include

Si se necesita que un fichero esté disponible, se puede usar require o require_once, que lanzan errores si no está disponible.

Pero si se usa include o include_once, y se quiere controlar el error, se debe comprobar la existencia del fichero.

```
<?php
if (file_exists('config.php')) {
        include 'config.php';
} else {
        // Manejar el error: registrar, mostrar mensaje amigable, etc.
        error_log('Archivo config.php no encontrado.');
        die('Error de configuración.'); // Finaliza el script
}
?>
```

PHP – Paths al incluir ficheros

Absolutos – Hacen referencia a la estructura del disco, no a la de la aplicación php. Son menos portables entre distintos servidores, porque todos los servidores deben tener la misma estructura de disco. Ejemplos:

```
include '/var/www/html/includes/cabecera.php'; // Ruta Linux
include 'C:\\web\includes\cabecera.php';// Ruta Windows
```

Relativos – Hacen referencia a la estructura de la aplicación php. Pueden ser relativos al script en ejecución, o a la raíz de la aplicación PHP, en cuyo caso empezarán con '/'. Más portables ente servidores. Ejemplos:

```
include '../../includes/inicio-sesion.php'; // Relativa al script actual
include '/includes/sesion.php'; // Relativa a la raíz de la aplicación
include 'fun.php'; // Relativa a script actual, en mismo directorio.
```

PHP – Paths al incluir – include_path

La directiva "include_path" permite configurar una serie de directorios en los que PHP buscará cuando se haga include de un fichero.

Esta directiva se puede configurar en el fichero php.ini:

```
include_path = ".:/usr/local/lib/php" ; PHP.ini
```

O se puede configurar en un script PHP en tiempo de ejecución:

```
$inicial = get_include_path()
set_include_path($inicial . PATH_SEPARATOR . 'otra/carpeta');
```

PHP, cuando encuentra un include o require, buscará en los directorios indicados en el include path.

Este mecanismo centraliza las rutas de inclusion.

PHP – Paths al incluir – __DIR__ y __FILE__

__DIR__ (con dos guiones bajos al principio y al final) es una constante que devuelve el directorio en el que se encuentra el fichero en el que se consulta la constante. Se puede usar para montar la ruta de includes:

```
include __DIR__ . "images" . nombreImagen;
```

__FILE__ devuelve el path del fichero en el que se consulta la constante.

En ambos casos, siempre devuelve una ruta absoluta en el , respecto al disco, no respecto a la aplicación.

Se puede obtener la ruta del directorio a partir de la del fichero, con la función "dirname":

```
$directorio = dirname(__FILE__); // Devuelve lo mismo que __DIR__
```

PHP – Otras formas de modularizar

Autoloading de clases:

- Carga automáticamente las clases utilizadas sin include / require.

Autoloading con composer:

- Sistema de configuración basado en json, similar a npm, o Maven

Uso de motores de plantillas, que simplifican la generación de HTML:

- Twig. El más popular. Symfony, Laravel o Drupal lo utilizan.
- Blade. El motor nativo en Laravel.
- Otros menos populares: Smarty, Plates, PHPTAL, Latte

PHP – Fichero php.ini

Es el principal fichero de configuración de PHP.

Controla cómo se comporta PHP en el servidor, y permite configurar distintas directivas para ajustar rendimiento, seguridad y funcionalidad.

¿Dónde se encuentra? Se puede:

- Ejecutar *php --ini*.
- En un script php, llamar a la función *phpinfo()*, que muestra toda la configuración de PHP. Buscar "*Loaded Configuration File*". O usar la función *php_ini_loaded_file()*.

Al ejecutar php, se busca el fichero .ini en distintos directorios. En la referencia de PHP está el orden de prioridad en la búsqueda:

https://www.php.net/manual/en/configuration.file.php

PHP – Fichero php.ini – Estructura

Es un archivo de texto plano, con directivas de configuración.

Cada directiva se escribe en una línea sigue la estructura:

directiva = valor

Por ejemplo:

memory_limit = 128M

Los comentarios se pueden escribir con ";" o con "#". Ejemplos:

```
; Esto es un comentario en .ini
# Esto es también un comentario
memory_limit = 128M ; El comentario puede tras la directiva
log_errors = Off # También con almohadilla
```

PHP – Fichero php.ini – Valores

Lógicos – On/Off o 1/0

display_errors = On

Numéricos – Pueden incluir unidades, como M para megabyte

max_execution_time = 30
post_max_size = 8M

Texto – En general, se puede escribir sin comillas

error_log = "/var/log/php_errors.log"

Listas – Algunas directivas admiten varios valores, en líneas separadas

extension = mbstring
extension = gd

PHP – Fichero php.ini – Ejemplos

; Muestra o no los errores en la salida
display_errors = Off

; Nivel de reporte de errores
error_reporting = E_ALL & ~E_NOTICE & ~E_DEPRECATED

; Límite de memoria
memory_limit = 256M

; Tiempo máximo de ejecución de un script
max_execution_time = 60

; Tamaño máximo de datos POST
post_max_size = 16M

PHP – Fichero php.ini – Ejemplos

; Tamaño máximo de archivos subidos
upload_max_filesize = 10M

; Ruta para guardar archivos de sesión
session.save_path = "/var/lib/php/sessions"

; Cargar extensiones de PHP
extension=mbstring
extension=gd
extension=mysqli

; Configuración de logging de errores
log_errors = On
error_log = "/var/log/php_errors.log"Confi