Envío de datos y archivos al servidor

DWES UD3

3.- Características básicas de PHP

Superglobales

PHP dispone de un conjunto de variables que se pueden utilizar **en cualquier ámbito** si necesidad de usar la palabra reservada **global**. Estas variables son **arrays** que contienen información interesante del sistema.

$_SERVER

Contiene información sobre el servidor

Gracias a la función **print_r** se puede mostrar todo el contenido de un array. Esta instrucción viene muy bien en tareas de depuración.

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

1.- Peticiones HTTP

Las peticiones HTTP son el medio mediante el cual los clientes web piden recursos a los servidores web.

Existen diferentes tipos de peticiones, el propio nombre de la petición nos indica la acción que se requiere sobre el recurso que se indica.

Las peticiones más habituales son los tipos:

- **GET**: se solicita un recurso específico del repositorio del servidor.
- **POST**: se envían datos para que los procese un recurso específico del servidor.

1.- Peticiones HTTP

Los tipos de peticiones HTTP existentes son los siguientes:

- **GET**
- **HEAD**
- **POST**
- **DELETE**
- **PUT**
- **PATCH**
- **TRACE**
- **CONNECT**
- **OPTIONS**

Ahora de momento solo usaremos los tipos **GET** y **POST.**
Más adelante en el curso, cuando se vean los servicios web API Restful se verán también los tipos:

PUT, PATCH y DELETE

2.- Variables superglobales de PHP

Variables predefinidas que están disponibles en **todos los ámbitos** del script.

Estas variables contienen diferente tipo de información: datos del propio servidor, datos del entorno, de las cookies…

$GLOBALS
$_SERVER
$_GET
$_POST
$_FILES
$_COOKIE
$_SESSION
$_REQUEST
$_ENV

2.- Variables superglobales de PHP

Contienen las variables que puede recibir un script php mediante los métodos que indican el nombre de la variable superglobal.

$_GET → datos en la URL o de un formulario con método GET
$_POST → datos de un formulario con método POST
$_COOKIE → cookies del cliente de la propia aplicación web

Junta en un solo array los datos de los tres anteriores

$_REQUEST

2.- Variables superglobales de PHP

Superglobales

Contiene la información de los archivos que se hayan mandado con el método POST.

$_FILES

Sirve para consultar las variables de sesión. Más adelante se explicará su uso.

$_SESSION

3.- Formularios web

Formularios web

Los formularios web son la herramienta que permite recoger datos introducidos por el usuario y procesarlos en el servidor.

Es importante elegir bien el método (**method**) con el que se enviará el formulario (**get** o **post**).

Recuerda que en el atributo **action** del formulario se tiene que indicar el script que recibirá los datos.

Para que un campo del formulario se envíe tiene que tener el atributo **name**.

3.- Formularios web

[programming tip]

El script que reciba los datos del formulario puede ser el mismo que el que muestra el propio formulario.

Se comprueba si existe alguna de las variables que deberían llegar desde el formulario y en ese caso se tratan los datos.

Si no existe dicha variable, entonces se mostrará el formulario.

3.- Formularios web

Formularios web

```
<form action="procesa.php" method="post">
        Nombre del alumno: <input type="text" name="nombre" id="nombre"><br>
        Apellidos del alumno: <input type="text" name="apellidos" id="apellidos"><br>

        Ciclo que cursa:<br>
        <input type="radio" name="ciclo" value="DAW"> Des. de ap. Web
        <br>
        <input type="radio" name="ciclo" value="DAM"> Des. de ap. Multiplataforma
        <br> <br>

                <input type="submit" value="Enviar">
</form>
```

3.- Formularios web

Formularios web

Archivo **procesa.php**:

```
<?php
   echo 'El alumno ';
   echo $_POST['nombre'] .' '. $_POST['apellidos'];
   echo '<br>Se encuentra cursando el ciclo: ';
   echo $_POST['ciclo'];
?>
```

3.- Formularios web

Formularios web

En el caso de que un **checkbox** pueda enviar varios valores hay que indicar en el atributo **name** que es un **array**.

```
<input type="checkbox" name="modulos[ ]" value="DWEC">
```

3.- Formularios web

Formularios web

```
<form name="input" action="#" method="post">
	Nombre del alumno: <input type="text" name="nombre"><br>
	Ciclos que cursa:<br>

	<input type="checkbox" name="modulos[ ]" value="DWES">
	Desarrollo web en entorno servidor<br>
	<input type="checkbox" name="modulos[ ]" value="DWEC">
	Desarrollo web en entorno cliente<br>
	<br>

	<input type="submit" value="Enviar">
</form>
```

3.- Formularios web

Formularios web

En ocasiones puede interesar que las variables que se envían sean un array:

```
<form name="input" action="#" method="post">
        Nombre: <input type="text" name="propio[nombre]"><br>
        Apellidos: <input type="text" name="propio[apellidos]"><br>
        Nombre: <input type="text" name="conyuge[nombre]"><br>
        Apellidos: <input type="text" name="conyuge[apellidos]"><br>
        <br>
        <input type="submit">
</form>
```

Ejercicio

Formularios web

Realizar los ejercicios 1 y 2 del boletín.

4.- Validación de datos

Formularios web

La validación de datos es muy importante y se debe realizar en 3 lugares:

- **Navegador**: mediante el uso de los tipos correctos en los campos **input** y el atributo **required**.

- **Cliente** (JavaScript): antes de que sean enviados con el fin de evitar sobrecarga en el servidor.

- **Servidor:** para evitar la suplantación de identidad y CSRF.Esto se realiza mediante un **input type="hidden"** y variables de sessión (se verán más adelante).

4.- Validación de datos

Formularios web

Es habitual que la misma página que muestra el formulario sea la que procese los datos que este envía. **action="#"**

Si todos los datos son correctos se realiza la acción que se debe.

En el caso de existir campos con errores, por ejemplo un nombre de persona con números, se vuelve a mostrar el formulario rellenando los campos con los valores que se introdujeron pero indicando qué campos contienen errores. Para esto se usa el atributo **value** de los **input** y **checked** de las **casillas de verificación**.

Para realizar esto último se usa la función **isset** para saber si al archivo llegan variables desde el método con el que se envía el formulario.

4.- Validación de datos

Formularios web - Validación de datos

Si se detecta una vulneravilidad, en vez de procesar los datos se redirige de nuevo al formulario:

```
header('Location: http://localhost.actividades/registro.php');
exit;
```

En caso contrario, proseguir validando campos mediante dunciones isset, is_numeric, strcmp... o con **expresiones regulares**.

Si en algún campo existe un error entonces recargar la pagina mostrando un mensaje informativo. Generalmente se rellenará de nuevo el formulario con los datos ya introducidos.

4.- Validación de datos

Expresiones regulares

Son patrones de caracteres que permiten comprobar si una cadena de texto se ajusta a un formato específico.

Algunos ejemplos son

- Que un nombre no tenga números.
- Que los campos tengan una longitud concreta.
- Que se siga un orden de números y caracteres determinado (DNI).

Las expresiones regulares se delimitan entre los caracteres **'/** y **/'**:

```
$expresion = '/[a-z]{4}/';          // 4 minúsculas
```

4.- Validación de datos

Expresiones regulares

Una vez definida la expresión regular, se puede comprobar si una cadena cumple con el patrón definido en la expresión regular con la función:

preg_match()

Esta función se usa de la siguiente manera:

preg_match($expresion, 'cadena de caracteres')

4.- Validación de datos

Expresiones regulares

Ejemplos de uso:

```
if(preg_match('/[a-zA-Z]{6}/', 'cadena de caracteres')) {
        …
}
```

```
$expresion = '/[a-zA-Z]{6}/';
if(!preg_match($expresion, $_POST['nombre'])) {
        // $_POST['nombre'] no coincide con el patrón
}
```

4.- Validación de datos

Expresiones regulares

La sintaxis para definir patrones en expresiones regulares en el lenguaje PHP se puede consultar en la documentación oficial:

https://www.php.net/manual/en/reference.pcre.pattern.syntax.php

Ejercicio

Formularios web

Realizar los ejercicios 3 y 4 del boletín.

5.- Tipos MIME

MIME → Multipurpose Internet Mail Extensions

Al principio del uso de internet, para el envío de contenido por mail se definió el **estándar MIME**. De esta manera se podía identificar el tipo de contenido que se enviaba.

Hoy en día los tipos MIME se utilizan para indicar el tipo de archivo que se envía por internet.

Cuando el servidor envía cualquier archivo (html, php, png, mp3…) lo primero que hace es enviar las cabeceras. En una de las cabeceras se indica el tipo de archivo:

estándar w3.org

5.- Tipos MIME

Por defecto cualquier script PHP genera código html el cual es texto, por lo que la cabecera que se envía será como la siguiente:

Content-type: text/html

Durante la carga completa de una página web además del archivos de texto HTML se envían otros archivos: css, javascript, imágenes…

Para cada uno de esos envíos el servidor indica en la cabecera el tipo MIME del archivo en cuestión de manera automática.

5.- Tipos MIME

A cada archivo se le asigna una cadena de texto que describe cómo es el contenido del archivo.

Esta cadena se forma de la siguiente manera: tipo/subtipo

Gracias a los tipos MIME tanto los clientes como los servidores tienen información sobre los archivos que van a intercambiar. Algunos ejemplos muy utilizados en el desarrollo de aplicaciones web son:

- text/css
- application/javascript

Lista completa de tipos MIME

6.- Subir ficheros al servidor

La manera de poder enviar ficheros al servidor desde la aplicación web es mediante el uso de formularios. Para poder enviar ficheros hay que tener en cuenta los siguientes aspectos:

- El método de envío del formulario siempre ha de ser **post**.
- El formulario ha de tener el atributo **enctype="multipart/form-data"**.
- El servidor tiene configurado un tamaño máximo de archivo.
- Los ficheros se almacenan en un directorio temporal y cuando acaba la ejecución del script que recibe los datos estos ficheros se eliminan.

6.- Subir ficheros al servidor

Para enviar ficheros se utilizan los **input** tipo **file** en el formulario:

```
<form action="subida.php" method="post" enctype="multipart/form-data">
        Selecciona el archivo a subir:
        <input type="file" name="archivo" id="archivo">
        <input type="submit" value="Enviar">
</form>
```

En un formulario se pueden enviar uno o varios archivos, para ello solo hará falta un campo input file por archivo.

6.- Subir ficheros al servidor

Para cambiar el tamaño máximo de subida se puede optar por dos opciones:

- En el archivo de configuración **php.ini**:
  upload_max_filesize = 2M

  En esa sección del archivo se pueden configurar otros parámetros para la subida de archivos al servidor.

- Añadiendo un campo oculto en el formulario indicando el tamaño en bytes:
  <input type="hidden" name="MAX_FILE_SIZE" value="1000000">

  El valor de MAX_FILE_SIZE no puede ser mayor al especificado en upload_max_filesize

6.- Subir ficheros al servidor

Una vez enviado uno o más archivos al servidor, desde el script que recibe los datos del formulario se puede usar el **array** superglobal **$_FILES** para acceder a la información de los archivos subidos.

El array $_FILES tendrá tantos elementos como archivos subidos desde el formulario.

Cada elemento del array será un array con la información de un archivo:

- tmp-name: ruta del archivo temporal en el servidor.
- name: nombre del archivo original (en el cliente).
- size: tamaño en bytes del archivo.
- type: tipo MIME del archivo (image/gif, text/plain...).
- error: código de error (UPLOAD_ERR_OK si se ha subido correctamente).

6.- Subir ficheros al servidor

Dado el siguiente formulario:

```
<form action="subida.php" method="post" enctype="multipart/form-data">
        Selecciona el archivo a subir:
        <input type="file" name="archivo" id="archivo">
        <input type="submit" value="Enviar">
</form>
```

En el archivo **subida.php** se puede acceder a los datos enviados de la siguiente manera:

```
        echo $_FILES['archivo']['tpm_name'];
```

6.- Subir ficheros al servidor

Fallos típicos

- No tener permiso sobre la ruta indicada en la directiva **upload_tmp_dir** en **php.ini**.
- La directiva **memory_limit** en php.ini tiene un valor muy pequeño o menor a **upload_max_filesize**.
- La directiva **max_execution_time** en php.ini tiene un valor bajo y la ejecución del script incluida la subida del fichero excede dicho tiempo.
- La directiva **post_max_size** en php.ini tiene un valor muy bajo, su valor debe ser mayor a upload_max_filesize (tamaño máximo archivo + resto de datos que se envían en el formulario).

Ejemplo

```
if ($_FILES['imagen']['error'] != UPLOAD_ERR_OK) { // Se comprueba si hay un error al subir el archivo
        echo 'Error: ';
        switch ($_FILES['imagen']['error']) {
                case UPLOAD_ERR_INI_SIZE:
                case UPLOAD_ERR_FORM_SIZE:      echo 'El fichero es demasiado grande';          break;
                case UPLOAD_ERR_PARTIAL:    echo 'El fichero no se ha podido subir entero';     break;
                case UPLOAD_ERR_NO_FILE:    echo 'No se ha podido subir el fichero';            break;
                default:        echo 'Error indeterminado.';
        }
        exit();
}

if ($_FILES['imagen']['type'] != 'image/gif') { // Se comprueba que sea del tipo esperado
        echo 'Error: No se trata de un fichero .GIF.';
        exit();
}
```

```
// Si se ha podido subir el fichero se guarda
if (is_uploaded_file($_FILES['imagen']['tmp_name']) === true) {
	// Se comprueba que ese nombre de archivo no exista
	$nombre = './subidas/'.$_FILES['imagen']['name'];
	if (is_file($nombre) === true) {
		$idUnico = time();
		$nombre = $idUnico.'_'.$nombre;
	}

	// Se mueve el fichero a su nueva ubicación
	if (!move_uploaded_file($_FILES['imagen']['tmp_name'], $nombre)) {
		echo 'Error: No se puede mover el fichero a su destino';
}
else
	echo 'Error: posible ataque. Nombre: '.$_FILES['imagen']['name'];
```

6.- Subir ficheros al servidor - SEGURIDAD

Como es fácil de deducir, el hecho de recibir archivos desde el cliente puede implicar riesgos de seguridad, por ello hay que controlar que todo funcione como se espera.

Primero se realiza la comprobación de errores **$_FILES[‘imagen’]['error']**, de esta manera sólo se continuará con el script en caso de que el archivos se haya subido de manera correcta.

A continuación se debe de comprobar que el tipo de archivo recibido sea el esperado **$_FILES[‘imagen’]['type']**.

Por último habrá que comprobar que el archivo realmente se ha subido y no se está accediendo a un archivo ya almacenado en el servidor.

6.- Subir ficheros al servidor - SEGURIDAD

A finales de 2000 existía un ataque que consistía en engañar al script que recibe el formulario para que cogiera un archivo local, accediendo así a información privada del servidor (/etc/passwd).

Para esto se creó la función **is_uploaded_file** que comprueba que el archivo se ha cargado desde HTTP POST.

Para almacenar el archivo en el servidor se puede usar la función **copy**, pero esta no asegura que el archivo que se mueve sea uno cargado, por ello se creó la función **move_uploaded_file**.

Además es conveniente usar la función **is_file** para asegurarse que no se está sobreescribiendo un archivo. Aunque hay en casos que es necesario sobreescribir (al actualizar la foto de perfil del usuario).

6.- Subir ficheros al servidor

Funciones del sistema de Archivos

Al trabajar con archivos en el servidor, es una buena ayuda conocer las funciones que permiten trabajar con el sistema de archivos (documentación).

Algunas funciones útiles:

- delete
- realpath(_ _FILE_ _)
- dirname
- is_dir
- rename
- mkdir
- rmdir

Ejercicio

Formularios y archivos

Realizar los ejercicios 5 y 6 del boletín.

7.- Tratamiento de imágenes en PHP

En la construcción de una aplicación web entran diversos elementos o medios, los llamados medias (multimedia), los más habituales son el texto, las imágenes y el vídeo.

El tratamiento del texto es algo sencillo y que ya se ha visto a lo largo del curso.

El tratamiento de las imágenes puede parecer que no sea necesario debido a que generalmente solo se muestran sin sufrir ningún tipo de modificación.

En determinados casos es interesante poder crear imágenes desde cero o incluso modificar las imágenes existentes antes de mostrarlas.

7.- Tratamiento de imágenes en PHP

Algunos ejemplos:

- Crear gráficos de barras.
- Añadir marca de agua a una imagen.
- Mostrar una versión a menor resolución.
- Guardar varias versiones de una imagen enviada por el usuario.

Existen diversas bibliotecas de funciones que se puede usar en PHP para el tratamiento de imágenes.

PHP dispone de una versión propia de la biblioteca **GD2** por lo que es habitual que se encuentre ya habilitada en cualquier instalación AMPP.

```
phpinfo()
```

7.- Tratamiento de imágenes en PHP - Librería GD2

La librería gráfica **GD2** está escrita en C.

Permite **crear** y **manipular** gráficos e imágenes.

Permite importar y exportar en diferentes tipos: **gif**, **png** y **jpg**.

GIF y PNG almacenan una matriz de puntos con su color concreto y comprimen el resultado (agrupando colores iguales continuos) para rebajar el peso del archivo.

JPG utiliza algoritmos complejos para comprimir (comprimiendo zonas de la imagen por separado).

7.- Tratamiento de imágenes en PHP - Librería GD2

Tipos MIME

Como se ha visto anteriormente en el tema, cada envío de archivo del servidor al cliente supone cambiar el tipo MIME de la cabecera de la petición.

Si se quiere usar un script PHP para la generación desde cero de imágenes lo primero que habrá que indicar en el script es el tipo de dato que se va a enviar cambiando la cabecera que se enviará:

```
<?php
header('Content-Type: image/png');
...
?>
```

7.- Tratamiento de imágenes en PHP - Librería GD2

Como ya se ha visto, existen diferentes tipos MIME, los más habituales cuando se generan imágenes son:

image/jpeg
image/gif
image/png
image/svg+xml

Lista MIME de Mozilla

7.- Tratamiento de imágenes en PHP - Librería GD2

Ejemplo de carga de imagen usando la librería GD2:

Código en index.php:

```
<img src="logo.php" alt="logo">
```

Código en logo.php

```
<?php
header('Content-Type: image/png');
$logo = imagecreatefrompng('logo.png’);
imageAlphaBlending($logo, true);
imageSaveAlpha($logo, true);
imagepng($logo);
imagedestroy($logo);
?>
```

7.- Tratamiento de imágenes en PHP - Librería GD2

En determinados casos puede ser interesante generar imágenes desde cero.

El proceso requiere de 4 pasos básicos:

- Crear lienzo sobre el que trabajar.
- Dibujar formas y/o imprimir texto en dicho lienzo.
- Generar la imagen final.
- Liberar los recursos.

7.- Tratamiento de imágenes en PHP - Librería GD2

```php
<?php
// Creación y configuración del lienzo
$alto = $ancho = 200;
$imagen = imagecreatetruecolor($ancho, $alto);
$blanco = imagecolorallocate($imagen, 255, 255, 255);
$azul = imagecolorallocate($imagen, 0, 0, 64);

// Se dibuja la imagen
imagefill($imagen, 0, 0, $azul);
imageline($imagen, 0, 0, $ancho, $alto, $blanco);
imagestring($imagen, 4, 50, 150, 'DWES', $blanco);

// Generación de la imagen.
header('content-type: image/png');
imagepng ($imagen);

// Liberamos la imagen de memoria.
imagedestroy($imagen);
?>
```

7.- Tratamiento de imágenes en PHP - Librería GD2

Crear imágenes

La función **imagecreatetruecolor** permite crear un lienzo en blanco de las medidas que se indiquen.

Si se quiere crear el lienzo a partir de una imagen existente se pueden usar las siguientes funciones indicando la ruta de la imagen:

- imagecreatefromgif
- imagecreatefromjpeg
- imagecreatefrompng
- imagecreatefromstring (detecta automáticamente el tipo, muy util si se obtiene la ruta de la imagen desde la base de datos)

De esta manera el lienzo tendrá el tamaño de la imagen que se carga.

7.- Tratamiento de imágenes en PHP - Librería GD2

Crear imágenes

Para poder pintar en la imagen hay que crear el color con el que se pintará:

color **imagecolorallocate** (imagen, rojo, azul, verde);

Se puede usar también colores con nivel de transparencia:

color **imagecolorallocateaphpa** (imagen, rojo, verde, azul, opacidad);

Los valores para los colores se expresan mediante la notación RGB:

0 a 255

Para la opacidad desde 0 (opaco) a 127 (transparente).

7.- Tratamiento de imágenes en PHP - Librería GD2

Crear imágenes

Una vez creado el lienzo y con un color seleccionado, se puede pintar en dicho lienzo usando las diferentes funciones existentes:

http://php.net/manual/es/book.image.php

Las funciones usadas en el ejemplo anterior:

```
imagefill($imagen, 0, 0, $azul);
    imageline($imagen, 0, 0, $ancho, $alto, $blanco);
    imagestring($imagen, 4, 50, 150, 'DWES', $blanco);
```

7.- Tratamiento de imágenes en PHP - Librería GD2

La función **imagestring**, que permite añadir texto a una imagen es muy limitada.

Si se necesita escribir texto en una imagen se recomienda usar la función **imagettftext** que permite usar fuentes descargadas en el servidor.

```
imagettftext(imagen, tamanyo, angulo, x, y, color, fuente, texto)
```

ejemplo:

```
imagettftext($imagen, 20, 0, 11, 21, $gris, 'arial.ttf', 'DWES');
```

7.- Tratamiento de imágenes en PHP - Librería GD2

Transformar imágenes

Suele ser interesante transformar las imágenes subidas por los usuarios:

- Poner marca de agua de la aplicación web.
- Guardar varias versiones/tamaños de la imagen.

7.- Tratamiento de imágenes en PHP - Librería GD2

```
$img_org = imagecreatefromgif("dwes.gif");

$ancho_dst = intval(imagesx($img_org) / 2);
$alto_dst = intval(imagesy($img_org) / 2);

$img_dst = imagecreatetruecolor($ancho_dst, $alto_dst);

imagecopyresized($img_dst, $img_org, 0, 0, 0, 0,
                                 $ancho_dst, $alto_dst,
                                 imagesx($img_org), imagesy($img_org));

header("Content-type: image/png");
imagepng($img_dst);


imagedestroy($img_org);
imagedestroy($img_dst);
```

7.- Tratamiento de imágenes en PHP - Librería GD2

También se puede usar la función **imagecopyresampled** que aplica un suavizado a los bordes lo cual obtiene una imagen de mayor calidad.

Si los tamaños no son proporcionales la imagen se verá distorsionada.

EL eje de coordenadas empiezan en la esquina superior izquierda.

Para rotar imágenes se usa la función:

resultado **imagerotate**(imagen, angulo, colorFondo)

7.- Tratamiento de imágenes en PHP - Librería GD2

Para cambiar el tamaño de una imagen se usa la función **imagescale**, a la que habrá que proporcionar las nuevas medidas, si solo se indica el ancho el reescalado será proporcional:

```
$imagen = imagescale ($imagen, 50);
```

En la documentación se pueden encontrar todas las funciones disponibles.

Ejemplo marca de agua

```
$marca = imagecreatefrompng('marca.png');
$marca = imagescale($marca, 50);
imagealphablending($marca, false);
imagesavealpha($marca,true);

$marcaX = imagesx($marca);
$marcaY = imagesy($marca);
imagefilter($marca, IMG_FILTER_COLORIZE, 0, 0, 0, 60);

$imagen = imagecreatefromjpeg('logo.jpg');
$destX = imagesx($imagen) - $marcaX - 5;
$destY = imagesy($imagen) - $marcaY - 5;

imagecopy($imagen, $marca, $destX, $destY, 0, 0, $marcaX, $marcaY);
header('content-type: image/jpeg');
imagejpeg($imagen);

imagedestroy($imagen);
imagedestroy($marca);
```

7.- Tratamiento de imágenes en PHP - Librería GD2

Todo lo visto hasta ahora se ha ejecutado mostrando la imagen en el cliente.

En ocasiones se puede transformar una imagen sin necesidad de mostrarla en el cliente, por ejemplo, cuando un usuario sube su foto de perfil el script encargado de guardar los datos puede redimensionar la imagen antes de guardarla y no mostrarla en el cliente.

Para guardar una imagen se utiliza la misma función que para mostrarla añadiendo la ruta destino:

```
imagejpeg($imagen, 'img/foto.jpg');
imagepng($imagen, 'img/foto.png');
```

Estas funciones se pueden usar añadiendo opciones (ver documentación).

Ejercicio

Formularios y archivos

Realizar los ejercicios 7 y 8 del boletín.