Práctica UD-4 Sesión-1

Objetivos	• Aprender conceptos básicos de desarrollo web con PHP.
Instrucciones de entrega	• No hay que entregar nada.

- Crea un nuevo proyecto en NetBeans (de tipo PHP->PHP Application) o en vuestro editor favorito con nombre "***PHPPracticaUD4Sesion1***".

Variables globales

- PHP proporciona unas variables predefinidas o variables integradas en todos los scripts:
    - *$GLOBALS*: hace referencia a todas las variables disponibles en el ámbito global.
    - *$_SERVER*: Información del entorno de ejecución y del servidor.
    - *$_GET*: variables HTTP GET.
    - *$_POST*: variables HTTP POST.
    - *$_FILES*: Variables de carga de archivos HTTP.
    - *$_REQUEST*: Variables de solicitud HTTP.
    - *$_SESSION*: Variables de sesión.
    - *$_ENV*: Variables de entorno.
    - *$_COOKIE*: Cookies HTTP.
- Estas variables predefinidas en PHP son "superglobales" o variables globales automáticas, lo que significa que están disponibles en todos los ámbitos a través de un script. No es necesario hacer una variable $global; para acceder a ellas dentro de funciones o métodos.
    - https://www.php.net/manual/es/reserved.variables.php

Ejemplo

- Crea un script php con nombre ***global.php*** con el siguiente contenido y ejecútalo:

```php
<?php
function test() {
    $foo = "variable local";

    echo '$foo en el ámbito global: ' . $GLOBALS["foo"] . "<br>";
    echo '$foo en el ámbito simple: ' . $foo . "<br>";
}

$foo = "Contenido de ejemplo";
test();
```

Formularios (GET y POST)

- Para el tratamiento de los formulario vamos a hacer uso de las dos variables predefinidas: $_GET y $_POST, que como hemos visto son superglobales y que forman arrays de claves y valores, donde las claves son los nombres de los inputs del formulario (atributo "name") y los valores son los datos de entrada de los usuarios.

- Como recordatorio aquí están los 10 tipos de elementos más importantes para los formularios HTML con PHP:

Elemento	Descripción
input type="text"	Caja de texto
input type="password"	Caja de texto donde se muestran asteriscos en lugar de los caracteres escritos
input type="checkbox"	Cajas seleccionables que permite escoger múltiples opciones
input type="radio"	Cajas seleccionables en grupos que sólo permiten escoger una opción
input type="submit"	Botón para enviar el formulario
input type="file"	Cajas de texto y botón que permite subir archivos
input type="hidden"	Elemento escondido. Especialmente útil para tokens de seguridad
option	Una opción posible dentro de un elemento element
select	Lista de opciones de elementos option
input type="textarea”	Texto multilínea

Ejemplo

- Crea un nuevo scripts php con nombre ***form.php*** con el siguiente contenido:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
 <p>Enviar formulario con GET:</p>
        <form action="formget.php" method="get">
            Nombre: <input type="text" name="nombre"><br>
            Email: <input type="text" name="email"><br>
            <input type="submit" value="Enviar">
        </form>
        <br>
        <br>
<p>Enviar formulario con POST:</p>
        <form action="formpost.php" method="post">
            Nombre: <input type="text" name="nombre"><br>
            Email: <input type="text" name="email"><br>
            <input type="submit" value="Enviar">
        </form>
    </body>
</html>
```

- Crea un nuevo scripts php con nombre ***formget.php*** con el siguiente contenido:

```
<?php
$nombre = isset($_GET["nombre"]) ? $_GET["nombre"] : "";
printf("Hola $nombre");
echo "<br>";
$email = isset($_GET["email"]) ? $_GET["email"] : "";
echo "Tu email es: $email";
```

- Crea un nuevo scripts php con nombre ***formpost.php*** con el siguiente contenido:

```
<?php
$nombre = isset($_POST["nombre"]) ? $_POST["nombre"] : "";
printf("Hola $nombre");
echo "<br>";
$email = isset($_POST["email"]) ? $_POST["email"] : "";
echo "Tu email es: $email";
```

- Pruebe el resultado navegando a la url:
    - http://localhost/PHPPracticaUD4Sesion1/form.php

Filtrado de datos

- PHP tiene algunas funciones que se usan habitualmente para verificar el contenido de las variables enviadas, esas funciones son
    - isset
    - empty
    - is_numeric

- Pero además se pueden utilizar la extensión de filtrado que nos da más poder y nos facilita las cosas:
    - https://www.php.net/manual/es/book.filter.php

- Las funciones que podemos utilizar son:
    - *filter_has_var*: Comprueba si existe una variable de un tipo concreto existe
    - *filter_id*: Indica el ID del nombre de filtro que se indique
    - *filter_input_array*: Obtiene variables externas y opcionalmente las filtra
    - *filter_input*: Toma una variable externa concreta por su nombre y opcionalmente la filtra
    - *filter_list*: Devuelve una lista de todos los filtros soportados
    - *filter_var_array*: Retorna múltiple variables y opcionalmente las filtra
    - *filter_var*: Filtra una variable con el filtro que se indique

- Junto con las constantes predefinidas:
    - https://www.php.net/manual/es/filter.constants.php

Ejemplo

- Crea un nuevo scripts php con nombre ***filterget.php*** con el siguiente contenido:

```php
<?php
if (!filter_has_var(INPUT_GET, 'email')) {
    echo "email no encontrado como parámetro GET";
} else {
    echo "email encontrado como parámetro GET";
}
```

- Pruebe el resultado navegando a la url:
    - http://localhost/PHPPracticaUD4Sesion1/filterget.php
    - http://localhost/PHPPracticaUD4Sesion1/filterget.php?email=hola@hola.es
    - http://localhost/PHPPracticaUD4Sesion1/filterget.php?email

Validación

- Todo formulario requiere de una validación para garantizar la exactitud de los datos. Como ejemplo, supongamos que tiene un campo de entrada para obtener la edad del usuario. Si un usuario escribe una cadena como "hola" allí y la envía y se guarda ese valor sin ninguna validación, su base de datos habrá guardado datos no válidos que pueden romper el sistema.

- Junto con la validación utilizaremos **funciones de limpieza** (en inglés sanitize) para adecuar los valores introducidos a nuestros formatos deseados, por ejemplo, evitar guardar caracteres en blanco para un nombre.

Ejemplo

- Crea un nuevo scripts php con nombre ***contactform.php*** con el siguiente contenido:

```php
<?php

// Funciones para filtrar las entradas del usuario
function filterName($field) {
    // Limpiar el nombre del usuario
    $sanitizeField = filter_var(trim($field), FILTER_SANITIZE_STRING);

    // Validate user name
    if (filter_var($sanitizeField, FILTER_VALIDATE_REGEXP,
array("options" => array("regexp" => "/^[a-zA-Z\s]+$/")))) {
        return $sanitizeField;
    } else {
        return FALSE;
    }
}

function filterEmail($field) {
    // Limpiar el correo electrónico
    $sanitizeField = filter_var(trim($field), FILTER_SANITIZE_EMAIL);

    // Validar el correo electrónico
    if (filter_var($sanitizeField, FILTER_VALIDATE_EMAIL)) {
        return $sanitizeField;
    } else {
        return FALSE;
    }
}

function filterString($field) {
    // Limpiar una cadena
```

```
    $sanitizeField = filter_var(trim($field), FILTER_SANITIZE_STRING);
    if (!empty($sanitizeField)) {
        return $sanitizeField;
    } else {
        return FALSE;
    }
}

// Definimos las variables y las inicializamos con la cadena vacía
$nameErr = $emailErr = $messageErr = "";
$name = $email = $subject = $message = "";

// Procesar los datos del formulario cuando es enviado
if (filter_input(INPUT_SERVER, 'REQUEST_METHOD') == "POST") {

    // Validar el nombre de usuario
    if (!filter_has_var(INPUT_POST, "name")) {
        $nameErr = "Por favor introduzca su nombre.";
    } else {
        $name = filterName(filter_input(INPUT_POST, "name"));
        if ($name == FALSE) {
            $nameErr = "Por favor introduzca un nombre válido.";
        }
    }

    // Validar correo electrónico
    if (!filter_has_var(INPUT_POST, "email")) {
        $emailErr = "Por favor introduzca su dirección de correo
electrónico.";
    } else {
        $email = filterEmail(filter_input(INPUT_POST, "email"));
        if ($email == FALSE) {
            $emailErr = "Por favor introduzca una dirección de correo
electrónico válida.";
        }
    }

    // Validar asunto
    if (!filter_has_var(INPUT_POST, "subject")) {
        $subject = "";
    } else {
        $subject = filterString(filter_input(INPUT_POST, "subject"));
    }

    // Validar comentario
    if (!filter_has_var(INPUT_POST, "message")) {
```

```
        $messageErr = "Por favor introduzca su comentario.";
    } else {
        $message = filterString(filter_input(INPUT_POST, "message"));
        if ($message == FALSE) {
            $messageErr = "Por favor introduzca un comentario válido.";
        }
    }

    // Comprobar los errores de entrada antes de enviar el correo
electrónico
    if (empty($nameErr) && empty($emailErr) && empty($messageErr)) {

        // Recipient email address
        $to = 'webmaster@example.com';

        // Crear las cabeceras del email
        $headers = 'From: ' . $email . "\r\n" .
                'Reply-To: ' . $email . "\r\n" .
                'X-Mailer: PHP/' . phpversion();

        // Mandar el email
        if (mail($to, $subject, $message, $headers)) {
             echo '<p class="success">¡Su mensaje se ha enviado
correctamente! Gracias.</p>';
        } else {
            echo '<p class="error">¡Lo sentimos no se puede enviar el
email!. Por favor inténtelo de nuevo</p>';
        }
    }
}
?>
<!DOCTYPE html>
<head>
    <meta charset="UTF-8">
    <title>Formulario de contacto</title>
    <style type="text/css">
        .error{ color: red; }
        .success{ color: green; }
    </style>
</head>
<body>
    <h2>Contacte con nosotros</h2>
    <p>Por favor rellene el siguiente formulario y pulser enviar.</p>
    <form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>"
method="post">
        <p>
```

```
            <label for="inputName">Nombre:<sup>*</sup></label>
            <input type="text" name="name" id="inputName" value="<?php
echo $name; ?>">
            <span class="error"><?php echo $nameErr; ?></span>
        </p>
        <p>
            <label for="inputEmail">Email:<sup>*</sup></label>
            <input type="text" name="email" id="inputEmail" value="<?php
echo $email; ?>">
            <span class="error"><?php echo $emailErr; ?></span>
        </p>
        <p>
            <label for="inputSubject">Asunto:</label>
            <input type="text" name="subject" id="inputSubject"
value="<?php echo $subject; ?>">
        </p>
        <p>
            <label for="inputComment">Mensaje:<sup>*</sup></label>
            <textarea name="message" id="inputComment" rows="5"
cols="30"><?php echo $message; ?></textarea>
            <span class="error"><?php echo $messageErr; ?></span>
        </p>
        <input type="submit" value="Enviar">
        <input type="reset" value="Reset">
    </form>
</body>
</html>
```

- Pruebe el resultado navegando a la url:
    - http://localhost/PHPPracticaUD4Sesion1/contactform.php

- El formulario lo hemos definido con el atributo que referencia al nombre de la misma página y también previene de hackeos:
    - *action=<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>*
        - *$ _SERVER ["PHP_SELF"]*: devuelve el nombre de archivo del script que se está ejecutando actualmente.
        - La función htmlspecialchars () convierte caracteres especiales en HTML. Esto significa que reemplazará los caracteres HTML cómo <y> por *<* y *&gt*;. Esto evita hackeos (de tipo Cross-site Scripting attacks) donde los atacantes exploten el código inyectando código HTML o Javascript en formularios.

Subida de archivos

- Además de un campo de selección de archivo ***input="file"***, el formulario de carga debe utilizar el método de publicación ***POST*** y debe contener un atributo ***enctype = "multipart / form-data"***. Este atributo garantiza que los datos del formulario se codifiquen como datos *MIME* de varias partes, que se requieren para cargar grandes cantidades de datos binarios como imágenes, audio, video, etc.

- Una vez que se envía el formulario, se puede acceder a la información sobre el archivo cargado a través de la matriz superglobal llamada ***$ _FILES***. En el caso del formulario del ejemplo que contiene un campo de selección de archivo (llamado *name="photo"*), podemos obtener los siguientes detalles a través de la ***matriz asociativa*** *$ _FILES ["photo"]*:
    - *$ _FILES ["photo"] ["name"]*: especifica el nombre original del archivo, incluida la extensión del archivo. No incluye la ruta del archivo.
    - *$ _FILES ["photo"] ["type"]*: especifica el tipo MIME del archivo.
    - *$ _FILES ["photo"] ["size"]*: especifica el tamaño del archivo, en bytes.
    - *$ _FILES ["photo"] ["tmp_name"]*: especifica el nombre temporal, incluida la ruta completa, que se asigna al archivo una vez que se ha subido al servidor.
    - *$ _FILES ["photo"] ["error"]*: especifica el código de estado o error asociado con la carga del archivo, p. Ej. será 0, si no hay error.

Ejemplo

- Crea una nueva carpeta llamada ***upload*** dentro del proyecto.
- Crea un nuevo archivo HTML con nombre ***uploadfileform.html*** con el siguiente contenido:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Formulario de subida de archivos</title>
    </head>
    <body>
        <form action="uploadfile.php" method="post"
enctype="multipart/form-data">
        <h2>Subir imagen</h2>
        <label for="fileSelect">Nombre archivo:</label>
        <input type="file" name="photo" id="fileSelect">
        <input type="submit" name="submit" value="Upload">
        <p><strong>Nota:</strong>Sólo archivos con extensión .jpg,
.jpeg, .gif, .png y con tamaño máximo 5 MB.</p>
    </form>
    </body>
</html>
```

- Crea un nuevo scripts php con nombre ***uploadfile.php*** con el siguiente contenido:

```php
<?php

// Comprobar que el formulario se ha enviado
if(filter_input(INPUT_SERVER, 'REQUEST_METHOD') == "POST"){
    // Check if file was uploaded without errors
    if(isset($_FILES["photo"]) && $_FILES["photo"]["error"] == 0){
        $allowed = array("jpg" => "image/jpg", "jpeg" => "image/jpeg",
"gif" => "image/gif", "png" => "image/png");
        $filename = $_FILES["photo"]["name"];
        $filetype = $_FILES["photo"]["type"];
        $filesize = $_FILES["photo"]["size"];

        // Comprobar la extensión del archivo
        $ext = strtolower(pathinfo($filename, PATHINFO_EXTENSION));
        if(!array_key_exists($ext, $allowed)){
            die("Error: Por favor seleccione un formato válido.");
        }

        // Comprobar que el tamaño del fichero es menor que 5MB
        $maxsize = 5 * 1024 * 1024;
        if($filesize > $maxsize){
            die("Error: El tamaño del archivo supera el límite
permitido.");
        }

        // Verify MYME type of the file
        if(in_array($filetype, $allowed)){
            // Check whether file exists before uploading it
            if(file_exists("upload/" . $filename)){
                echo $filename . " ya existe.";
            } else{
                move_uploaded_file($_FILES["photo"]["tmp_name"],
"upload/" . $filename);
                echo "El fichero ha sido enviado correctamente.";
            }
        } else{
            echo "Error: Ha ocurrido un problema al subir el archivo.
Por favor inténtelo de nuevo.";
        }
    } else{
        echo "Error: " . $_FILES["photo"]["error"];
    }
}
```

- Pruebe el resultado navegando a la url:
    - http://localhost/PHPPracticaUD4Sesion1/uploadfileform.html

Redirección

- La redirección en PHP se puede hacer usando la función header (). Para configurar, una simple redirección simplemente crea un archivo index.php en el directorio desde el que desea redireccionar con el siguiente contenido:
    - <?php header("Location: http://www.redirect.to.url.com/"); ?>
    - <?php header("Location: anotherDirectory/anotherFile.php"); ?>

- Si estamos haciendo una redirección de la página completa el header irá al principio del script y también se recomienda usar siempre las funciones *die* o *exit* (son equivalentes) después del *header*. Por ejemplo:

```php
<?php
header("Status: 301 Moved Permanently");
header("Location: http://www.ejemplo.es");
exit;
```

Ejemplo

- Crea un nuevo scripts php con nombre ***redirect.php*** con el siguiente contenido:

```php
<?php
header("Status: 301 Moved Permanently");
header("Location: index.php");
die;
```

Ejercicios:

1. Divide la página ***contactform.php*** en dos: ***contactform.php*** con el contenido del formulario html y ***contactprocessform.php*** que contenga el procesado del formulario.

2. Activa el envío de correo electrónico para que pueda enviarse correctamente.
    a. https://meetanshi.com/blog/send-mail-from-localhost-xampp-using-gmail/

3. Añade un campo para poder subir un archivo al mensaje y envíalo también adjunto en el correo electrónico:

4. Modifica el archivo ***uploadfile.php*** para que muestre todos los archivos que hay en la carpeta *upload* en forma de enlace para poderse descargar.

Recursos

- https://www.php.net/manual/es/book.filter.php
- https://www.w3schools.com/php/php_ref_filter.asp
- https://www.tutorialrepublic.com/php-tutorial/php-filters.php
- https://diego.com.es/formularios-en-php
- https://ejemplos.net/ejemplos-de-formularios-php/
- https://jonathanmelgoza.com/blog/subida-de-archivos-en-php/
- https://www.tutorialrepublic.com/php-tutorial/php-file-upload.php
- https://www.bluehost.com/help/article/php-redirect