Módulo 4: Entrada/Salida. Formularios. Ficheros

Objetivos

- Conocer el mecanismo de paso de información entre páginas a través de formularios y sus particularidades.
- Gestionar el almacenamiento y recuperación de información de manera externa a la página mediante el uso de ficheros.

Contenidos

En este módulo se expondrán, en suma, algunos de los métodos que permiten a una página web transferir información a otro, y guardarla para una posterior obtención.

Índice de apartados

Módulo 4: Operatividad con formularios

En este apartado, vamos a ver como operar con formularios y, en particular, nos detendremos a examinar el tratamiento que éstos hacen de la información que se quiere transmitir.

Como prerequisito, resultaría conveniente recordar todo lo concerniente a HTML relacionado con su diseño, esto es, aquellas etiquetas implicadas en la creación de un formulario.

Aconsejamos a tal efecto la siguiente página:

- http://www.desarrolloweb.com/articulos/647.php?manual=21

Un ejemplo para empezar ...

El siguiente ejemplo podría ser el formulario más sencillo que puede darse. Como se puede observar, contiene un campo de texto (en el que se nos sugiere introducir nuestro nombre) y un botón de envío.

Cópialo, guárdalo en tu carpeta de prácticas con el nombre *practica41.htm* y pruébalo en el navegador.

NOTA: para el correcto funcionamiento del paso de información a través de los formularios es conveniente que la directiva REGISTER_GLOBALS del fichero de configuaración "php.ini" está a "On".

```
<html>
<body>
<form method="post" action="practica41_resp.php">
<p> Introduce tu nombre:
<input name="nombre" type="text">
<br>
<input type="submit" name="enviar" value="enviar" >
</p>
</form>
</body>
</html>
```

>> Observaciones:

- un formulario (como el anterior) hace de cada objeto "input" una variable, que tendrá aparejada un dato. En nuestro caso, se generarán dos variables:
  - *nombre* (perteneciente al botón tipo "text")
  - *enviar* (perteneciente al botón tipo "submit")
- la etiqueta "form" contiene dos parámetros importantes:
  - **method:** define la modalidad de envío de la información. Los valores posibles son *POST* o *GET*; en el primer caso, los valores se transmiten de manera oculta (no aparecen en la barra de direcciones del navegador); en el segundo caso, aparecería, por ejemplo: *http://localhost/processa.php?nom=jaume&curs=1&grup=A*

- La clausula **action**: indica el nombre fichero receptor de los datos del formulario.

Como hemos sugerido, la acción de pulsar el botón de envió del formulario desencadena la transmisión de ciertos a datos a una página receptora, que deberá ser código PHP. Este podría ser un ejemplo (cópialo, guárdalo en tu carpeta de prácticas con el nombre *practica41_resp.php* y pruébalo en el navegador).

```
<?php

/*-------------------------------------------------------------
* Módulo: 4 Práctica: 1 Fichero: practica41_resp.php
* Descripción: Ejemplo formulario (1)
* Pre condi.:
* Post cond.:
-------------------------------------------------------------*/


// Exponemos 2 posibilidades: la primera usa el array $_POST
$op1 = $_POST['nombre'];

echo "Hola, ".$op1."!<br><br>";
echo "¿Te gusta el PHP?";

// Esta segunda no lo utiliza (si tienes activado el uso de variables globales
$op2 = $nombre;

echo "Hola, ".$op2."! (este nombre no aparecerá)<br><br>";
echo "¿Te gusta el PHP?";
?>
```

Este script muestra el contenido de la variable "nombre" originada en el formulario de dos formas. En primer lugar, recurre al array **$_POST** -que abordaremos a continuación-; en segundo lugar, vuelca su contenido en la variable "op", para mostrar finalmente su contenido por pantalla.

***Arrays* $_POST i $_GET**

En versiones de PHP posteriores a la 4.2.0, existen dos formas de acceso a las variables transmitidas a través de un formulario:

- A través de **variables globales**: cada nombre de variable del formulario (name="...") está disponible como variable global ($codi). Este es el sistema tradicional y el más simple de implementar, pero también más inseguro, por lo cual lo desaconsejamos (además, solo funcionará si en el " php.ini" la directiva *register_globals* está activada).
- A través de tres arrays asociativos que PHP genera en este proceso:

Arrays	Explicación
_POST o **HTTP_POST_VARS**	Array asociativo que contiene las variables pasadas por el método POST
_GET o **HTTP_GET_VARS**	Array asociativo que contiene las variables

	pasadas por el método GET

La manera de acceder a estos arrays sería, por ejemplo: **$_POST['nombre']** donde "nombre" sería el nombre de la variable y, por tanto, equivalente a *$nombre,* teniendo en ambos casos el valor introducido en el formulario.

Sugerimos la ejecución de ambas páginas con el parámetro *action* ajustado a "GET", a fin de apreciar la diferencia.

Una variante ...

Veamos ahora una variante al ejemplo anterior.

Cópialo, guárdalo en tu carpeta de prácticas con el nombre *practica41_b.php* y pruébalo en el navegador.

```
<?

/*-------------------------------------------------------------
* Módulo: 4 Práctica: 1b Fichero: practica41_b.php
* Descripción: Ejemplo formulario (2)
* Pre condi.:
* Post cond.:
-------------------------------------------------------------*/

?>

<html>
<body>

<form method="post" action="<? echo $_SERVER["PHP_SELF"]; ?>" >
<p> Introduce tu nombre:
<input name="nombre" type="text">
<br>
<input type="submit" name="enviar" value="enviar" >
</p>
</form>
</body>
</html>

<?
if(isset($_POST["enviar"])){
   echo "Hola, ".$_POST["nombre"]."!<br><br>";
   echo "¿Te gusta el PHP?"; }
?>
```

>> Observaciones:

- en este caso, lo que antes eran dos páginas (una, la "emisora" de información, html, y la otra , la receptora de la información, php) han quedado fundidas en una única, almacenada como .php . Por tanto, todo la funcionalidad queda resuelta en una única página. Para ello, en el parámetro action insertamos un trozo de código PHP en el que mostramos el valor de la variable interna **$PHP_SELF** que contiene el nombre del fichero actual interpretado (cabe observar como código HTML y PHP se combinan para este cometido).

- la función isset() determina si una variable dada está definida. Con ello, nos aseguramos que "venimos del formulario", ya que lo que estamos comprobando es que el botón "enviar" haya sido pulsado y, por tanto, que la variable haya sido generada.

Variables Predefinidas.

A partir de PHP 4.1.0, el método preferido para recuperar variables externas es mediante las superglobales. Antes de este punto, la gente recaía en register_globals o las matrices largas predefinidas en PHP ($HTTP_*_VARS). A partir de PHP 5.0.0, las matrices de tipo "long" de variables predefinidas, se pueden desactivar con la directiva register_long_arrays.

Para más información visitad esta página: http://es2.php.net/manual/es/reserved.variables.php

Un ejemplo más completo

Veamos ahora un ejemplo con un formulario más completo.

Copia el código y guárdalo en tu carpeta de prácticas con el nombre *practica42.htm* .

```
<html>
<head>
<title>Practica42.htm</title>
</head>

<body>
<h2>Formulario de ejemplo (3): </h2>
<form name="form1" method="post" action="practica42_resp.php">
<p>Nombre:
<input name="nombre" type="text" id="nombre" maxlength="20">
</p>
<p>Apellido:
<input name="apellido" type="text" id="apellido" maxlength="20">
</p>
<p>Selecciona un deporte:
<select name="deporte" id="deporte">
<option>futbol</option>
<option>basket</option>
<option>tenis</option>
<option>rugby</option>
</select>
</p>
<p>Sexo:</p>
<p>
<input name="sexo" type="radio" value="m">
Masculino</p>
<p>
<input name="sexo" type="radio" value="f">
Femenino</p>
<p>
<input name="conducir" type="checkbox" id="conducir" value="checkbox">
¿Te gusta conducir? </p>
<p>Aficiones:</p>
<p>
<textarea name="aficiones" cols="50" rows="5" id="aficiones"></textarea>
</p>
```

```
<input type="submit" name="enviar" value="enviar" >

<p> </p>
</form>
<p> </p>
<p> </p>
</body>
</html>
```

A continuación, se muestra un posible código de tratamiento de los datos enviados por el formulario anterior. Cópialo, guárdalo en tu carpeta de prácticas con el nombre *practica42_resp.php* y pruébalo en el navegador junto con su formulario.

```
<?

/*-------------------------------------------------------------------
* Módulo: 4 Práctica: 2 Fichero: practica42_resp.php
* Descripción: Ejemplo formulario (3)
* Pre condi.:
* Post cond.:
-------------------------------------------------------------------*/

// Comprobamos si venimos del formulario

if(!isset($_POST["enviar"]))
echo "No hay datos";
else {

// Dependiendo del tipo de objeto, utilizamos una
// estructura de control diferente

echo "Tu nombre es ".$_POST["nombre"]." ".$_POST["apellido"]."<br>";

if($_POST["conducir"])
echo "Has visto el anuncio de BMW <br>";
else
echo "No tienes un BMW <br>";

if($_POST["sexo"]=="m")
echo "Eres un hombre de pelo en pecho<br>";
else
echo "Eres toda una mujer <br>";

switch($_POST["deporte"]){

case "futbol": echo "Eres futbolero<br>";
break;
case "tenis": echo "Eres tenista<br>";
break;
case "basket": echo "Eres del Pamesa<br>";
break;
case "rugby": echo "Eres un bruto<br>";
break;

}

if($_POST["aficiones"]){
echo "Tus aficiones son:<br>";
```

```
echo nl2br($_POST["aficiones"]); }
else
echo "No tienes aficiones";
}

echo "<p><a href=\"practica42.htm\">Volver al formulario</a></p>"

?>
```

>> Observaciones:

- el script anterior no representa demasiadas dificultades; cada zona de código se ocupa de cada una de las variables recibidas del formulario, atendiendo al tipo de objeto de formulario que las genera.
- la función **nl2br()**, inserta saltos de línea HTML antes de cada salto de línea, buscando respetar la disposición de cada línea del "textarea" original.
- por último, tenemos un enlace de vuelta a la página del formulario.

Envío de ficheros desde un formulario

En este última sección, vamos a ver cómo podríamos hacer que una página web pudiera dar la posibilidad de "subir" un fichero seleccionado por el usuario al servidor.


Para ello, partiremos del siguiente código, que debes copiar, guardar como *practica43.htm*.

```
<HTML> <HEAD>
<TITLE>Carga de ficheros</TITLE>
</HEAD>
<BODY>
<FORM ENCTYPE="multipart/form-data"    ACTION="practica43_resp.php"
METHOD="POST">
 <INPUT TYPE="hidden" NAME="MAX_FILE_SIZE" VALUE="1024000">
<INPUT TYPE="file" NAME="mifichero">
<INPUT TYPE="submit" VALUE="Enviar">
</FORM>
</BODY>
</HTML>
```

>> Observaciones:

- El formulario anterior incluye algunas etiquetas novedosas:

  - En la etiqueta FORM: **< FORM enctype="multipart/form-data"..>** que indica que el formulario está integrado por contenidos de diferente tipo, no solo texto plano.
  - Un campo oculto que indica el tamaño máximo del archivo a enviar: **< input type="hidden" name="MAX_FILE_SIZE" value="1024000" >**

    NOTA: a ***MAX_FILE_SIZE*** no se le puede dar un valor mayor que el valor que se haya especificado en la directiva upload_max_filesize en el "php.ini" . Por defecto se tiene un límite de 2 MB.

- Un campo con el nombre y ubicación para el archivo: **< input type="file" name="mifichero" >**

Para el uso de este tipo de formularios se añaden una serie de variables que operan sobre ciertas características del archivo a transferir:

Variable	Descripción
$nombre_archivo o $nombre_archivo_**tmp**	Nombre del fichero temporal que se utiliza para almacenar en el servidor el archivo recibido.
$nombre_archivo_**name**	Nombre original del fichero en la máquina cliente.
$nombre_archivo_**size**	Tamaño del archivo en bytes.
$nombre_archivo_**type**	Tipo MIME del archivo (un ejemplo podría ser "image/gif").

Es decir, para nuestro caso, tendremos una variable **$mifichero** desmembrada en 4 variables más, cada una con su "extensión" (**_tmp,_name**, **_size** y **_type**) .

Pero como no podemos utilizar estas variables directamente por no tener a on la directiva de variables globales, utilizaremos la matriz **$_FILES** que contendrá estos valores:

Variable	Descripción
$_FILES['mifichero']['name']	Nombre original del fichero en la máquina cliente.
$_FILES['mifichero']['type']	Tipo MIME del archivo (un ejemplo podría ser "image/gif").
$_FILES['mifichero']['size']	Tamaño del archivo en bytes.
$_FILES['mifichero']['tmp_name']	Nombre del fichero temporal que se utiliza para almacenar en el servidor el archivo recibido.
$_FILES['mifichero']['error']	Es el código de error generado en la carga del fichero. 0 significa todo OK.

Veamos en el siguiente ejemplo cómo podríamos operar con ellas (cópialo, guárdalo como *practica43_resp.php* y ejecútalo junto al formulario anterior).

```
<?php

/*-------------------------------------------------------------------
* Módulo: 4 Práctica: 3 Fichero: practica43_resp.php
* Descripción: Ejemplo formulario (2)
* Pre condi.:
* Post cond.:
-------------------------------------------------------------------*/

// visualizamos el contenido de las variantes de "mifichero"

echo "Nombre: <B>".$_FILES['mifichero']['tmp_name']."</B>";
echo "<BR>";
echo "Tipo: <B>".$_FILES['mifichero']['type']."</B>";
echo "<BR>";
echo "Nombre original: <B>".$_FILES['mifichero']['name']."</B>";
echo "<BR>";
echo "Tamaño: <B>".$_FILES['mifichero']['size']."</B>";
echo "<BR>";
```

```
// En la siguiente línea indicamos la ruta donde guardaremos
// el fichero "temporal" (adáptala a tu caso)

$ruta = "./carpeta/".$_FILES['mifichero']['name'];
echo "<BR>";
echo "Ruta y nombre : <B>".$ruta;
echo "<BR>";
echo "<BR>";

// Comprobamos si podemos subir el fichero
// si es así, lo copiamos a su destino
// sino, damos el pertienente aviso por pantalla

if(move_uploaded_file($_FILES['mifichero']['tmp_name'],$ruta))
     echo "El archivo se ha subido con éxito";
else
   {
   echo "No pudo transferir el fichero: " . $_FILES['mifichero']['name']."<BR>";
   echo "Error: " . $_FILES['mifichero']['error'];
   }
// Si el archivo es demasiado grande, no se habrá subido

if($_FILES['mifichero']['error']==2) echo "Archivo demasiado grande";

?>
```

>> Observaciones:

- en las primera líneas, mostramos el contenido de las diferentes "extensiones" del la matriz **$_FILES['mifichero']**, recibida del formulario. Hay que diferenciar entre el nombre del fichero original y el nombre del fichero temporal . El primero hace referencia al nombre sin más; el segundo es un nombre para el fichero temporal transferido, el cual hay que renombrar y guardar en el lugar que decidamos de nuestro disco duro. Esto es así porque, el fichero transferido es temporal y se perderá si no es guardado.

- definimos una variable **$mifichero_def** en el cual definimos la ruta del directorio donde alojaremos nuestro fichero transferido.

- "subimos" el fichero al servidor mediante la **función move_uploaded_file()** , comprobando si se da algún error en el proceso.

- por último, comprobamos si el valor de la variable **$_FILES['mifichero']['error']** es **2**, ya que en ese caso habremos excedido el tamaño permitido y no se habrá efectuado la transferencia.

Un álbun de fotos: carga de archivos a una página web.

La idea es ofrecer a los usuarios un formulario HTML con un campo tipo file que nos permitirá cargar archivos.

- Una vez cargado el archivo, PHP se vuelve a activar y comprobará que el archivo sea correcto.
- Solo se podrán utilizar archivos gráficos en formato GIF o JPEG.
- El tamaño máximo permitido será de 200 KBytes (200.000 bytes).
- Todas las imagenes se almacenarán en una carpeta llamada imagenes.
- Una vez cargado se recorrerá la carpeta de imagenes mostrando cada una de ellas.

Veamos el siguiente ejemplo cópialo, guárdalo como *practica44.php* y ejecútalo.

```
<?

/*-------------------------------------------------------------
```

```
* Módulo: 4 Práctica: 4 Fichero: practica44.php
* Descripción: Carga de formularios
* Pre condi.:
* Post cond.:
-----------------------------------------------------------------*/

?>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>Cargar un archivo</title>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
</head>
<body>
<h1>Mi álbum de fotos en línea</h1>
<h3>Cargar archivo</h3>
<form action='<?php echo $_SERVER['PHP_SELF'] ?>' method="post"
enctype="multipart/form-data">
<input type="file" name="archivo">
<input type="submit" name='submit' value="Cargar archivo">
</form>
<?php
$ruta = "imagenes/"; // Indicar ruta

if (isset($_FILES['archivo']) && $_FILES['archivo']['size'] > 0) {
   $tamanyomax = 200000; // Indicar tamaño en bytes
   $nombretemp = $_FILES['archivo']['tmp_name'];
   $nombrearchivo = $_FILES['archivo']['name'];
   $tamanyoarchivo = $_FILES['archivo']['size'];
   $tipoarchivo = GetImageSize($nombretemp);

   if ($tipoarchivo[2] == 1 || $tipoarchivo[2] == 2) { // GIF o JPG?
      if ($tamanyoarchivo <= $tamanyomax) { // Archivo demasiado grande?
         if (move_uploaded_file($nombretemp, $ruta . $nombrearchivo)) {
            echo "<p>El archivo se ha cargado <b>con éxito</b>.
            Tamaño de archivo: <b>$tamanyoarchivo</b> bytes,
            Nombre de imagen: <b>$nombrearchivo</b><br></p>";
         } else {
            echo "<p>No se ha podido cargar el archivo.</p>";
         }
      } else {
         echo "<p>El archivo tiene más de <b>$tamanyomax bytes</b> y
         es demasiado grande.</p>";
      }
   } else {
      echo "<p>No es un archivo GIF
      o JPG válido.</p>";
   }
   echo "<form action='{$_SERVER['PHP_SELF']}' method='post'>
        <input type='submit' value='OK'></form>";
}
$filehandle = opendir($ruta); // Abrir archivos
while ($file = readdir($filehandle)) {
   if ($file != "." && $file != "..") {
      $tamanyo = GetImageSize($ruta. $file);
      echo "<p><img src='$ruta$file' $tamanyo[3]><br></p>\n";
   }
}
closedir($filehandle); // Fin lectura archivos
?>

</body>
</html>
```

Si lo que queremos es mostrar la anchura máxima de la imagen (por ejemplo 600 píxeles) modificaremós ligeramente la página y utilizaremos la función getImageSize() y su valor índice 0.

```
<?php
$ruta = "imagenes/"; // Indicar ruta

if (isset($_FILES['archivo']) && $_FILES['archivo']['size'] > 0) {
   $anchomax = 600;
   $nombretemp = $_FILES['archivo']['tmp_name'];
   $nombrearchivo = $_FILES['archivo']['name'];
   $tamanyoarchivo = $_FILES['archivo']['size'];
   $tipoarchivo = GetImageSize($nombretemp);

   if ($tipoarchivo[2] == 1 || $tipoarchivo[2] == 2) { // GIF o JPG?
      if ($tipoarchivo[0] <= $anchomax) { // Archivo demasiado ancho?
         if (move_uploaded_file($nombretemp, $ruta . $nombrearchivo)) {
            echo "<p>El archivo se ha cargado <b>con éxito</b>.
            Tamaño de archivo: <b>$tamanyoarchivo</b> bytes,
            Anchura: <b>$tipoarchivo[0]</b>
            Nombre de imagen: <b>$nombrearchivo</b><br></p>";
         } else {
            echo "<p>No se ha podido cargar el archivo.</p>";
         }
      } else {
         echo "<p>El archivo tiene una anchura superior a <b>$anchomax píxeles</b> y
         es demasiado ancho.</p>";
      }
   } else {
      echo "<p>No es un archivo GIF
      o JPG válido.</p>";
   }
   echo "<form action='{$_SERVER['PHP_SELF']}' method='post'>
        <input type='submit' value='OK'></form>";
```

Por último será recomendable que no pueda cualquiera subir un archivo al albún, para ello anadiremos un campo en el formulario que solicite el password, este password lo guardaremos en un archivo llamado password.inc.php que lo ubicaremos en una carpeta no accesible desde la web, casi siempre existe por defecto en servidores externos y se llama ../cgi-bin.

Una vez comprobado el password pasaremos a cargar el archivo:

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
 <title>Cargar un archivo</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<body>
<h2>Cargar archivo</h2>
<form action="<?php echo $_SERVER["PHP_SELF"]; ?>" method="post">
Contraseña: <input type="password" name="pass">
<input type="submit" value="Enviar">
</form>
<?php
$login = false;
//include("../cgi-bin/password.inc.php");
include("./password.inc.php");
if (isset($_POST['pass']) && $_POST['pass'] == $pw) {
   $login = true;
} else if (isset($_GET['formavis']) && $_GET['formavis'] == "zuxrkk") {
   $login = true;
}
if ($login) {
echo <<<FORMULARIO
<h1>Mi álbum de fotos en línea</h1>
<h3>Cargar archivo</h3>
```

```
<form action='{$_SERVER['PHP_SELF']}?formavis=zuxrkk' method="post"
enctype="multipart/form-data">
<input type="file" name="archivo">
<input type="submit" name='submit' value="Cargar archivo">
</form>
FORMULARIO;

$ruta = "./imagenes/"; // Indicar ruta

if (isset($_FILES['archivo']) && $_FILES['archivo']['size'] > 0) {
        $anchomax = 600;
        $nombretemp = $_FILES['archivo']['tmp_name'];
        $nombrearchivo = $_FILES['archivo']['name'];
        $tamanyoarchivo = $_FILES['archivo']['size'];
        $tipoarchivo = GetImageSize($nombretemp);

        if ($tipoarchivo[2] == 1 || $tipoarchivo[2] == 2) { // GIF o JPG?
            if ($tipoarchivo[0] <= $anchomax) {
                if (move_uploaded_file($nombretemp, $ruta . $nombrearchivo)) {
                    echo "<p>El archivo se ha cargado <b>con éxito</b>.
                Tamaño de archivo: <b>$tamanyoarchivo</b> bytes,
                Ancho: <b>$tipoarchivo[0]</b>
                Nombre de imagen: <b>" . "$nombrearchivo</b><br></p>";
                } else {
                    echo "<p>No se ha podido cargar el archivo</p>";
                }
            } else {
                echo "<p>El archivo tiene una anchura superior a <b>$anchomax píxeles</b> y
            es demasiado ancho.</p>";
            }
        } else {
        echo "<p>No es un archivo GIF
        o JPG válido.</p>";
        }
        echo "<form action='{$_SERVER['PHP_SELF']}?formavis=zuxrkk' method='POST'>
          <input type='submit' value='OK'></form>";
    }
    $filehandle = opendir($ruta); // Abrir archivos
    while ($file = readdir($filehandle)) {
        if ($file != "." && $file != "..") {
            $tamanyo = GetImageSize($ruta . $file);
            echo "<p><img src='$ruta$file' $tamanyo[3]><br></p>\n";
        }
    }
    closedir($filehandle); // Fin lectura archivos

}

?>
</body>
</html>
```

Envío y validación de formularios.

En este apartado veremos como enviar el contenido de los formularios por correo electrónico. En conjunción con el servicio SMTP instalado en el servidor (en el caso de linux suele ser Sendmail) se enviará la información.

En primer lugar crearemos un mini servicio de envío, después el servicio de envío universal y finalmente lo ampliaremos.

La función mail().

Su sintaxis es la siguiente:

mail("e-mail del distanatario", "asunto", "mensaje", "De: e-mail del remitente")

Es posible utilizar variables en lugar de las cadenas indicadas en la sintaxis.

En caso de que el envío sea satisfactorio, la función devuelve el valor TRUE, en caso contrario el valor devuelto será FALSE.

El mini sistema de envío:

Mostraremos un formulario que solicita el correo electrónico del destinatario y un campo textarea con el contenido del mensaje, así como un botón para realizar el envío. Si este se realiza correctamete se indicará con un mensaje, en caso contrario dara un mensaje de error, (cópialo, guárdalo en tu carpeta de prácticas con el nombre *minimail.php* y pruébalo en el navegador).

```
<html>
<head>
<title>Mini-Correo</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<link rel="stylesheet" type="text/css" href="../css/nuevo.css">
</head>
<body>
<h1>Mini-Correo</h1>
<p>Envíeme un mensaje de e-mail</p>
<form action="<?php echo $_SERVER["PHP_SELF"]; ?>" method="post">
Su dirección de correo electrónico:
<input type="text" name="mail">
<br>
Qué quiere decirme:<br>
<textarea name="mensaje" cols="50" rows="5" wrap="soft">
</textarea>
<br>
<input type="submit" value="Enviar mensaje">
</form>
<?php
if (isset($_POST["mail"]) && $_POST["mail"] != "")
{
if(mail("tu@correo.com", "Tiene correo nuevo", "$mensaje", "De:".$_POST["mail"])) {
        echo "<p>¡Gracias! Su mensaje ha sido enviado.</p>\n";
       }
```

```
   else {
echo "<p>Por desgracia, se ha producido un error al enviar.</p>\n";
}
}
?>
</body>
</html>
```

Si lo que queremos es un sistema de envío de formularios que envíe todos sus formularios, independientemente de cómo estén construidos y de cuántos campos tenga, necesitaremos utilizar los vectores asociativos para recorrer la totalidad de campos del formulario (para ello será necesario dar un nombre a todos los campos del formulario), (cópialo, guárdalo en tu carpeta de prácticas con el nombre *minimail2.php* y pruébalo en el navegador).

```
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8">
<title>Valoración de formularios</title>
</head>
<body>
<?php

/* LAS SIGUIENTES LÍNEAS SE PUEDEN EDITAR */
$destinatario="tu@correo.com";
$asunto="Escriba aquí el texto del mensaje";
$mensaje="Se han insertado los siguientes datos:\n\n";
/* FINAL DE EDICIÓN */

/* El bucle lee los pares atributo-valor: */
foreach($_POST as $name=> $value) {
/* Todos los datos se almacenan en $mensaje: */
$mensaje.="$name=$value\n"; // Versión abreviada para la concatenación
}

/* ¡Enviar! Preparar el campo de mail en el formulario */
if (isset($_POST["mail"]) && $_POST["mail"]!="") {
 // ¿Campo mail establecido y no vacío?
 /* Se activa la función de envío mail() */
 if(mail($destinatario, $asunto, $mensaje, "De: ".$_POST["mail"])) {
/* ¿Ha tenido éxito la función mail()? El usuario verá las siguientes líneas */
 echo "<h1>¡Gracias por sus comentarios!</h1>\n";
 echo "<p>Su mensaje ha sido enviado.</p>\n";
}
/* En caso contrario, se muestra un mensaje de error: */
else {
 echo "<h1>Desgraciadamente, no se ha podido enviar su mensaje.</h1>\n";
}
} // cerrar la función externa if

/* else si no se ha establecido la variable $mail: */
else {
 echo "<h1>Por favor, escriba su dirección de correo  electrónico.</h1>\n";
}
?>
</body>
</html>
```

Esta página php recibirá en el vector $_POST con todos los campos del formulario que llamo a esta página, se van concatenando en la variable $mensaje con el objetivo de envialos por correo electrónico.

Pero será necesario que la página que llame a esta tenga un formulario con un campo llamado mail (este campo será utilizado para saber de donde venimos), y por lo tanto daremos un mensaje de error en los casos de que el correo electrrónico del destinatario no existe o no se realice correctamente el envio y un mensaje de confirmación si el correo se envia correctamente,

Veremos una última versión del programa de envio de formulario por correo electrónico, en esta versión solicitaremos el nombre, los apellidos, el correo del destinatario y el cuerpo del mensaje.

Mostraremos el correo antes de confirmar el envio y en caso de no estar conforme podremos volver atrás, modificar y volver a confirmar el envio, (cópialo, guárdalo en tu carpeta de prácticas con el nombre *minimail3.php* y pruébalo en el navegador).

```
<html>
<head>
<title>El formulario para enviar su opinión</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<link rel="stylesheet" type="text/css" href="../css/nuevo.css">
</head>
<body>
<h1>El formulario para enviar su opinión</h1>
<?php
// ¿Botón submit vacío, 0 o no definido?
if (empty($_POST["submit"])) {
?>
<h3>1. Escriba sus datos</h3>

<form action="<?PHP echo $_SERVER["PHP_SELF"]; ?>" method="post">
Nombre: <input type="text" name="nombre"><br>
Apellido: <input type="text" name="apellido"><br>
E-mail: <input type="text" name="mail"><br><br>
Su opinión:<br>
<textarea name="feedback" cols="50" rows="5" wrap="soft">
</textarea><br><br>
<input type="reset" name="reset" value="xXx Borrar datos">
<!-- importante: ¡asignar el nombre submit al botón de envío! -->
<input type="submit" name="submit" value="Enviar ahora --&gt;">
</form>

<?php
}
// En caso contrario, es decir si el botón de submit existe
else {
?>
<h3>2. Vuelva a comprobar sus datos</h3>

<?php
foreach($_POST as $key => $value) {
// repasa todos los campos del formulario
if (empty($value)) {  // ¿valor vacío, 0 o no definido?
?>
<p>Por favor, rellene <b>todos los campos</b>.</p>
<form>
<input type="button" value="&lt;-- Volver al formulario"
onclick="javascript:history.back()">
</form>

<!-- Consejo de seguridad, visible sólo para usuarios sin JavaScript: -->
<noscript>Utilice el botón ATRÁS del navegador para volver</noscript>

<?php
exit; // termina el bucle y el programa.
```

```
}
}

echo "<p>Hola <b>".$_POST['nombre'].$_POST['apellido']."</b><br>\n";
echo "Su dirección de correo electrónico es <b>".$_POST["mail"]."</b>!<br>\n";

echo "Ha introducido la siguiente información:<br>\n";

// Eliminar las barras invertidas y mantener los saltos de línea para mostrar el contenido:
echo "<i>" . stripslashes(nl2br($_POST["feedback"])) . "</i><br><br>\n";
echo "¿Son correctos estos datos?</p>\n";

// Los datos introducidos por el usuario se escriben en un campo:
$mensaje=$_POST["nombre"].$_POST["apellido"].", ".$_POST["mail"]." ha escrito\n".$_POST
["feedback"];
$mensaje=htmlspecialchars($mensaje); // Enmascarar los caracteres especiales de HTML
$mensaje=stripslashes($mensaje); // Eliminar las barras invertidas

// Crear un nuevo formulario para reenviarlo al script de envío
echo "<form action=\"mail.php\" method=\"post\">\n";

// Truco: mostrar los datos de los campos ocultos del formulario:
echo "<input type=\"hidden\" name=\"mail\" value=\"".$_POST["mail"]."\">\n";
echo "<input type=\"hidden\" name=\"mensaje\" value=\"$mensaje\">\n";
echo "<input type=\"button\" value=\"&lt;-- No, por favor corríjalo\" ";
echo "onclick=\"javascript:history.back()\">\n";

// Concesión para los que desactiven JavaScript:
echo "<noscript>Vuelva atrás utilizando el botón ATRÁS.</noscript>\n";
echo "<input type=\"submit\" name=\"mailsender\" ";
echo "value=\"Perfecto --&gt;\"></form>\n";

}
?>

</body>
</html>
```

El fichero mail.php será el encargado de realizar el envío final:

```
<html>
<head>
<title>Mini-Correo</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<link rel="stylesheet" type="text/css" href="../css/nuevo.css">
</head>
<body>
<h1>Se envían los datos</h1>

<?php
if (isset($_POST["mail"]) && $_POST["mail"] != "") {
if(mail("clea@iescamp.com", "Tiene correo nuevo",$_POST["mensaje"], "De:".$_POST["mail"]))

{
echo "<p>¡Gracias! Su mensaje ha sido enviado.</p>\n";
}
else {
echo "<p>Por desgracia, se ha producido un error al enviar.</p>\n";
}
}
?>
</body>
</html>
```

Gestión de Ficheros

En este apartado vamos a trabajar con ficheros. Anteriormente, fuimos capaces de comprobar la afinidad existente entre PHP y C en tanto que, por ejemplo, su sintaxis general o sus estructuras de control presentaban un alto grado de similitud. En el caso de la gestión de ficheros, PHP cuenta con un cuerpo de funciones base similar al existente en C, respetando incluso la sintaxis de éstas.

Por lo tanto, comenzaremos haciendo un repaso de las más importantes, al que sumaremos algunas específicas de PHP, al tiempo que lo ilustraremos con ejemplos concretos.

Funciones más importantes

- **fopen(archivo,modo):** función de apertura de fichero que devuelve una valor numérico de tipo *integer* con la referencia al archivo abierto. Esta función dispone de diversos modos de apertura del fichero para lectura/escritura.
  - **r** Sólo lectura. El puntero al comienzo del archivo.
  - **r+** Lectura y escritura; apuntador al inicio del archivo.
  - **w** Sólo escritura, si no existe el archivo lo crea, si existe lo convierte en fichero de longitud cero y el apuntador se coloca en el inicio.
  - **w+** Lectura y escritura, si no existe el archivo lo crea, si existe lo convierte en fichero de longitud cero y el apuntador se coloca en el inicio.
  - **a** Modo append, sólo escritura, si no existe lo crea. Si existe, el apuntador se coloca al final del fichero.
  - a+ Modo append, lectuta y escritura, si no existe lo crea. Si existe, el apuntador se coloca al final del fichero.
- **fclose(indicador_archivo):** función que cierra un archivo valiéndose de su referencia en la apertura. Devuelve TRUE si el fichero se cerró correctamente, FALSE en caso contrario.
- **file_exists(fichero):** función que devuelve TRUE si el archivo especificado existe o FALSE en caso contrario.
- **fgets(indicador_archivo,longitud):** función que devuelve una cadena de como mucho longitud - 1 bytes leídos del fichero apuntado por el indicador_archivo. La lectura acaba cuando son leídos (longitud - 1) bytes, cuando se llega a una nueva línea (el caracter de nueva línea se incluye en el valor devuelto), o cuando se llega a un EOF (lo que ocurra primero).
- **fputs(indicador_archivo,cadena):** función que escribe una cadena en el fichero indicado.
  - El fichero debe estar previamente abierto
  - la función devuelve TRUE si se ha escrito con éxito y FALSE en caso contrario.
- **feof(indicador_archivo)**: función que comprueba si hemos alcanzado el final de fichero, devolviendo TRUE en tal caso.
- **flock(indicador_archivo, modo)**: bloquea o desbloquea el fichero para que no pueda ser abierto en modo escritura.

El siguiente ejemplo combina algunas de las funciones anteriores. Cópialo, guárdalo en tu carpeta de prácticas con el nombre *practica43_2.php* y pruébalo en el navegador.:

```php
<?
/*-------------------------------------------------------------------
* Módulo: 4 Práctica: 3 Fichero: practica43_2.php
* Descripción: Ejemplo funciones de ficheros (I)
* Pre condi.:
* Post cond.:
-------------------------------------------------------------------*/


$nombre="prueba.txt"

if(file_exists($nombre)){
    if($ref=fopen($nombre,"r"))
          while(!feof($ref)){
              $linea=fgets($ref,11);
              echo $linea;}
    }
    else echo "El archivo no existe";

fclose($ref);

?>
```

>> Observaciones:

- en la primera instrucción "if", comprobamos que el fichero $nombre existe con la ayuda de la función file_exists.
- de ser así, abrimos el fichero en modo lectura con la función fopen, para ...
- ... acto seguido, extraer once caracteres de cada una de las líneas del fichero de que conste el fichero mediante la función fgets hasta que alcancemos su final, detectado por la función feof .
- por último, cerramos el fichero abierto mediante la función fclose

Otras **funciones** que cabría tener presente serían:

- **readfile(fichero)**: función que visualiza el contenido de un fichero por la salida estándar (consola), preservando su disposición.
- **file():** Si queremos acceder a todo el contenido de un fichero y operar con él, podemos usar la función file(). Esta función lee el fichero línea por línea y devuelve un array (un elemento por línea).

El siguiente ejemplo combina utiliza la función **file.** Cópialo, guárdalo en tu carpeta de prácticas con el nombre *practica44.php* y pruébalo en el navegador:

```php
<?

/*-------------------------------------------------------------------
* Módulo: 4 Práctica: 4 Fichero: practica44.php
* Descripción: Ejemplo formulario (1)
```

```
* Pre condi.:
* Post cond.:
-----------------------------------------------------------------*/

$nombre="prueba.txt";

if(file_exists($nombre)){

// Con "file" generamos el array "texto"
// Cada línea dl fichero será un elemento del array

     $texto=file($nombre);

// Mostramos cada una de las líneas del array
// apoyándonos en las funciones 'list' y 'each'

     while(list($linea,$contenido)=each($texto))
          echo ($linea+1)." --> ".$contenido."<br>";
}

else
     echo "El archivo no existe";

?>
```

>> Observaciones:

- La función **file** nos permite volcar el contenido del fichero a un array de tantos elementos como líneas tenga el fichero.
- En el bucle "while" extraemos uno a uno, en cada iteración, los elementos del array $texto mediante la función each. Acto seguido, la función **list** se encarga de separar, por una lado, el índice del elemento, el cual es guardado en la variable $linea, y, por otro, el contenido del elemento, que es guardado en la variable $contenido . El contenido de ambas variables, $linea y $contenido , son mostrados en el navegador.
- Como se puede observar, en ningún momento hemos necesitado abrir y cerrar el fichero para poder acceder a su contenido, lo cual supone una ventaja.