Recoger datos del usuario (I)

PHP

1. INTRODUCCIÓN

La ventaja fundamental de utilizar una tecnología de programación de páginas dinámicas, como PHP, es que el resultado que recibe el usuario que solicita una página puede estar en función de los detalles que ha proporcionado.

La página se construye en el momento de la solicitud con el contenido personalizado, lo que evita tener almacenadas muchas páginas estáticas en su lugar.

Sin embargo, para todo ello tenemos que ver cómo recoger la información que nos facilita el usuario y que le solicitamos para conocer lo que realmente desea recibir.

En estos casos se suelen aplicar dos enfoques:

- Utilizar hipervínculos que apuntan a una única página pero proporcionando *"información adicional"* para que el servidor pueda personalizarla.
- Utilizar formularios HTML donde el usuario introducirá los datos necesarios y los enviará al servidor.

Independientemente de cómo se envíe la información al servidor, este tiene que ser capaz de recogerla y trabajar con ella. Para ello, PHP proporciona una serie de variables conocidas como **superglobales**. Ahora veremos cómo trabajar con ellas.

2. LA DIRECTIVA ELIMINADA REGISTER_GLOBALS

PHP proporciona una serie de arrays predefinidos con mucha información del **contexto** en el que se están ejecutando las páginas web. Así, podemos encontrar información sobre:

- El propio **servidor** web en el que están guardadas las páginas web.
- Información sobre **cookies**.
- Información transmitida mediante un **formulario**.
- Información transmitida en la propia **dirección URL** de la página web.
- Información almacenada en la **sesión** de cada usuario.
- Etc.

Toda esta información está disponible, sin necesidad de hacer nada, en cualquier código PHP, a través de los arrays **$_GET**, **$_POST**, **$_COOKIE**, etc.

Sin embargo, hasta la versión **4.2** de PHP no hacía falta indicar el nombre de estos arrays para poder acceder a la información enviada por el usuario porque la directiva de configuración **register_globals** tenía el valor predeterminado **On**. Esto, junto al hecho de que no sea necesario declarar las variables, representaba mucha flexibilidad pero también un importante resquicio en la seguridad de las aplicaciones php.

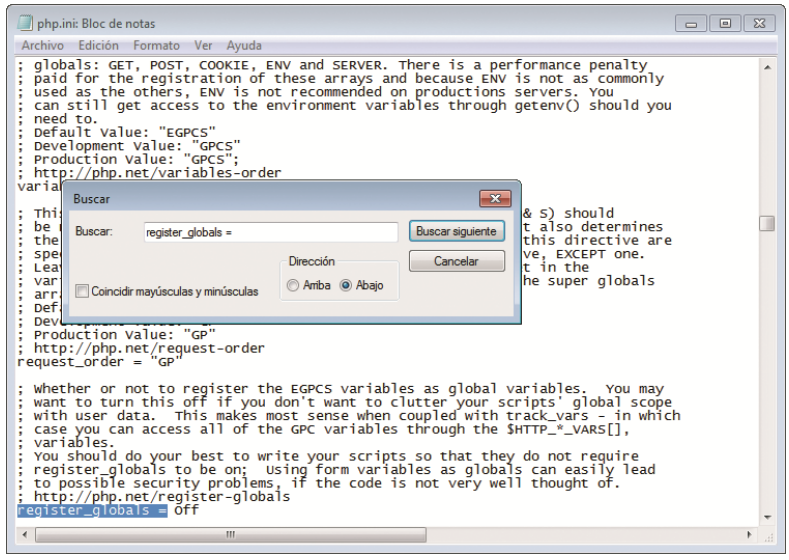
Por ejemplo, si el usuario escribía en un cuadro de texto de un formulario llamado **nombre**, nuestro código php era capaz de acceder a dicha información tan fácilmente como escribiendo el identificador **$nombre**.

A partir de dicha versión, la configuración inicial de PHP cambió y se estableció que era obligatorio indicar el nombre del array del que queremos obtener la información, es decir, que el valor predeterminado de **register_globals** pasaba a ser **Off**.

Actualmente la directiva **register_globals** ha sido eliminada, aunque podemos seguir encontrándola en el archivo de configuración **php.ini**. Esto quiere decir que si la directiva **register_globals** tiene el valor **On**, el intérprete de PHP producirá un error.

Por lo tanto, aquel código PHP que confía en que la directiva register_globals tiene el valor On debe ser reescrito si se actualiza a una versión moderna de PHP.

Sin embargo, si tiene el valor Off, no lo producirá aunque realmente ya no sirve para nada esta directiva a partir de la versión **PHP 5.3**, en la que quedó obsoleta y a partir de la versión **PHP 5.4**, en la que fue eliminada. En la figura siguiente vemos el archivo de configuración php.ini, que el paquete XAMPP instala en la carpeta **c:\xampp\php** (en otras instalaciones podría cambiar su ubicación).



Como te comentaba, la directiva **register_globals** es la que establecía si podemos utilizar el nombre de las variables del contexto, como la información enviada mediante un formulario, sin indicar dónde se almacenan o no.

En cualquier instalación de PHP es muy importante que esta directiva tenga el valor **Off**, aunque ya sabes que si utilizas una versión moderna esto será obligatorio en cualquier caso, ya que ha sido eliminada.

En la figura anterior vemos que sigue apareciendo esta directiva en el archivo **php.ini** que proporciona el paquete XAMPP. Realmente no es necesaria ya pero comprobamos que aparece con el valor **Off**, por lo que no producirá ningún error de ejecución.

Si trabajas con una instalación de PHP anterior a PHP 5.3, debes asegurarte de que **register_globals** tiene el valor **Off** aquí. Cámbialo si es necesario.

Por lo tanto, si quieres acceder a la información almacenada en algún campo de un formulario, deberás indicar expresamente dónde está almacenada. Lo mismo para otra información importante del contexto donde se está ejecutando la aplicación web, como iremos viendo.

3. EL ARRAY $_GET

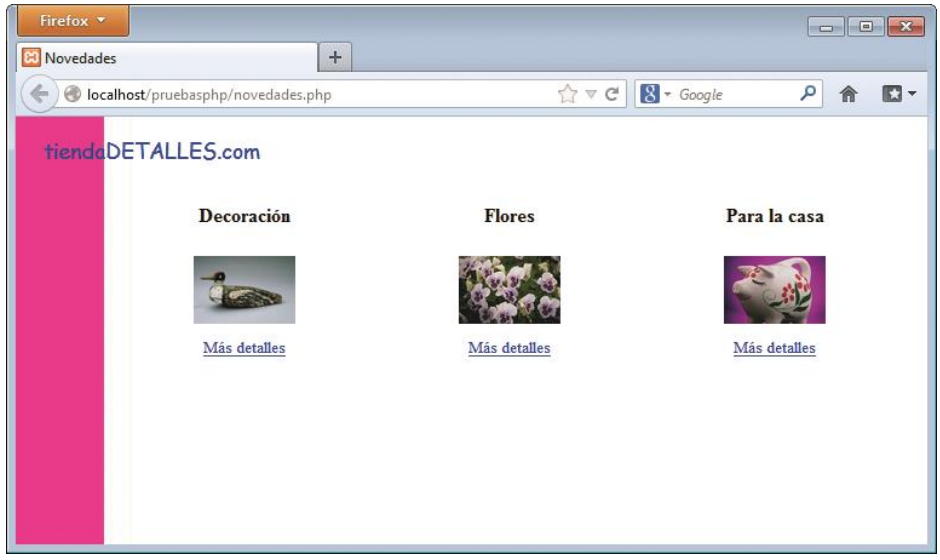
Como se ha indicado, sin necesidad de establecer ninguna configuración especial o escribir código adicional, la información de contexto de la aplicación la tenemos disponible a través de una serie de arrays **superglobales**.

El primer array de este tipo que vamos a estudiar es **$_GET**, que da acceso a la información de la página enviada al servidor mediante el método **Get**.

Este método de envío de una página web tiene la característica fundamental de que la información necesaria viene dada explícitamente en la URL o dirección de la propia página.

Veamos un ejemplo para que quede claro. Fíjate en la página web de la figura siguiente.

Se trata de una página sencilla correspondiente a los productos *"novedad"* de una tienda virtual en Internet. Hemos bautizado esta tienda con el nombre ficticio **tiendaDETALLES.com**.



Bien, cuando el usuario acceda a esta página verá que aparecen tres productos nuevos en el catálogo de la tienda: uno de decoración, otro de la categoría ramos de flores y otro de productos para la casa.

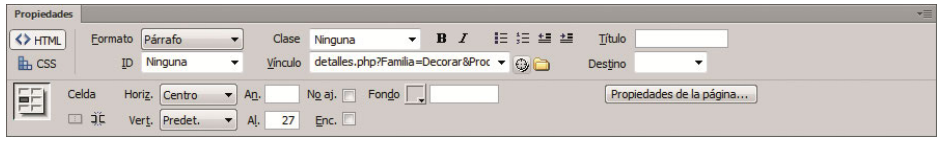
Si le interesa conocer algún detalle adicional de estos productos, pulsará en el hipervínculo "**Más detalles**" correspondiente.

Este texto no es más que un hipervínculo que apunta a una página donde encontrará los detalles adicionales del producto en cuestión.

Podríamos crear una página por cada producto y escribir la URL correspondiente en cada uno de estos hipervínculos.

Pero es mejor tener solo una página física y crearla con el contenido adecuado cuando así nos la soliciten.

Para ello, deberás establecer el vínculo o URL a la que apunta el hipervínculo. Utiliza el Inspector de propiedades de Dreamweaver, como puedes ver en la figura siguiente. Este detalle se establece en el campo **Vínculo**.



Así pues, el hipervínculo apunta a **detalles.php**, pero adicionalmente a esta página, tenemos que pasar información específica del producto del que quiere detalles el usuario:

detalles.php?Familia=Decorar&Producto=Cisne

Es muy importante no escribir ningún espacio en blanco en ambos lados del igual en esta información adicional que estamos pasando a la página **detalles.php**.

Además, vemos que se utiliza el carácter **?** para separar la parte de la URL correspondiente a la página destinataria de esta información adicional y **&** para separar cada clave que se le pasa.

En este caso estamos informando de que el usuario desea obtener detalles del producto **Cisne**, cuya familia es **Decorar**.

Fíjate que esto lo estás estableciendo al escribir la URL completa a la que apunta el hipervínculo. A estos detalles es a los que nos dará acceso después el array **$_GET**.

Lo mismo haríamos para los otros dos hipervínculos:

detalles.php?Familia=Flores&Producto=Ramo04

detalles.php?Familia=Hogar&Producto=Hucha

Fíjate que los tres hipervínculos apuntan a la misma página pero con información distinta.

La primera parte de nuestro trabajo ya está hecha. Ahora hace falta escribir el código necesario en la página **detalles.php** para recoger esta información y construir el contenido adecuado para el usuario que la ha solicitado. Para ello, utilizaremos el array superglobal **$_GET** en la página **detalles.php**.

Esta página es muy sencilla, ya que en ella simplemente he incluido un marcador de imagen, cierto texto y un cuadro donde poner la descripción del producto.

Fíjate que todos estos detalles se rellenarán en función del producto que haya elegido el usuario a través del hipervínculo correspondiente.

Veamos el código PHP necesario para ello:

```
<?php
$familia = $_GET["Familia"];
$producto = $_GET["Producto"];
switch ($producto)
{
     case "Cisne":
           $imagen = "images/Decoracion/Deco01.jpg";
           $descripcion = "Figura de porcelana china que
            representa un cisne. Ideal como sobremesa.";
           $pvp = 421.0;
           break;
     case "Ramo04":
           $imagen = "images/Flores/Flores04.jpg";
           $descripcion = "Ramo de flores en tonos
                          claros para decoración.";
           $pvp = 59.45;
           break;
     case "Hucha":
           $imagen = "images/Home/Home02.jpg";
           $descripcion = "Para ahorrar... qué mejor que
                          esta simpática hucha.";
           $pvp = 32.0;
           break;
}
?>
```

La clave de todo esto está en ver cómo se recoge la información adicional que se envía junto a la URL de la página.

```
$familia = $_GET["Familia"];
$producto = $_GET["Producto"];
```

Aquí está la forma de acceder a dicha información. Como se ha indicado, vendrá dada en la misma URL de la página web, por lo que utilizaremos el array **$_GET** indicando a qué clave queremos acceder.

Fíjate, que se trata de un array asociativo porque podemos utilizar el mismo nombre que se indica en la URL.

Por lo tanto, en la variable **$familia** tendremos la familia del producto y en la variable **$producto** el código de ese producto.

Con esos detalles, será sencillo personalizar la página que tenemos que enviar al usuario. Fíjate que en función del valor **$producto**, estableceremos otras tres variables:

- **$imagen**: donde indicamos la imagen a mostrar en la página de detalles.
- **$descripcion**: con el texto explicativo sobre el producto.
- **$pvp**: con el precio de venta al público.

Estos detalles son distintos para cada uno de los tres productos que son "novedad" en el catálogo de nuestra **tiendaDETALLES.com**.

Pues bien, ahora simplemente incorporaremos esos valores en la página resultante:

```
<body>
<p>Familia de productos: <?php echo "<b>$familia</b>"; ?></p>
<p><img name="producto" src=<?php echo "\"$imagen\""; ?>>
  PVP: <?php echo "<b>$pvp</b>"; ?></p>
<form name="form1" method="post" action="">
  <textarea          name="descripcion"          cols="80"
id="descripcion"><?php echo $descripcion; ?></textarea>
</form>
</body>
```

Por ejemplo, en el caso de la familia del producto, imprimiremos el valor de la variable **$familia** en negrita.

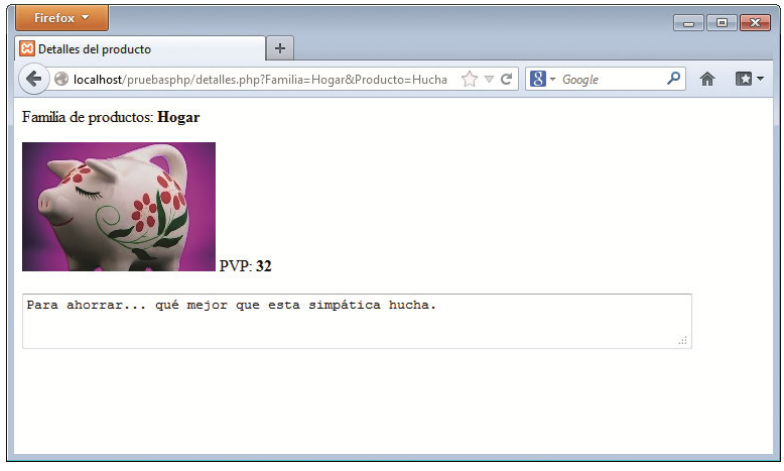
En el caso de la imagen a mostrar no es tan directo porque tenemos que incluir las comillas dobles mismas del atributo **src** de la etiqueta **<img>**.

Como tenemos el problema de que la misma cadena se especifica entre comillas dobles, tendremos que *"escaparlas"* con una barra inclinada hacia la izquierda delante:

```
<img name="producto" src=<?php echo "\"$imagen\""; ?>>
```

No es difícil, simplemente tienes que recordar la necesidad de anteponer una barra inclinada \ cuando quieras incluir comillas dobles dentro de una cadena rodeada por comillas dobles.

Como puedes ver en la figura siguiente, una vez hemos obtenido la información necesaria, particularizamos la página de forma muy sencilla, escribiendo el contenido adecuado.



La información que aparece en la propia URL de la página se conoce como query string. Por lo tanto, debes saber que cuando se habla de "query string" se está indicando información suministrada mediante el método Get del envío de páginas al servidor.

Observa la dirección URL de la página web anterior, donde aparece la información adicional que ha hecho posible que se genere el contenido adecuado.

Como puedes ver, el método es bastante sencillo, pero tiene el serio inconveniente de que el usuario puede ver la información en la URL de la página e incluso podría modificarla intentando obtener un contenido distinto al que debe obtener.

Esto nos hace pensar en la necesidad de disponer de un método distinto para pasar información confidencial o importante. Veremos a continuación que es posible mediante el método **Post**.

4. EL ARRAY $_POST

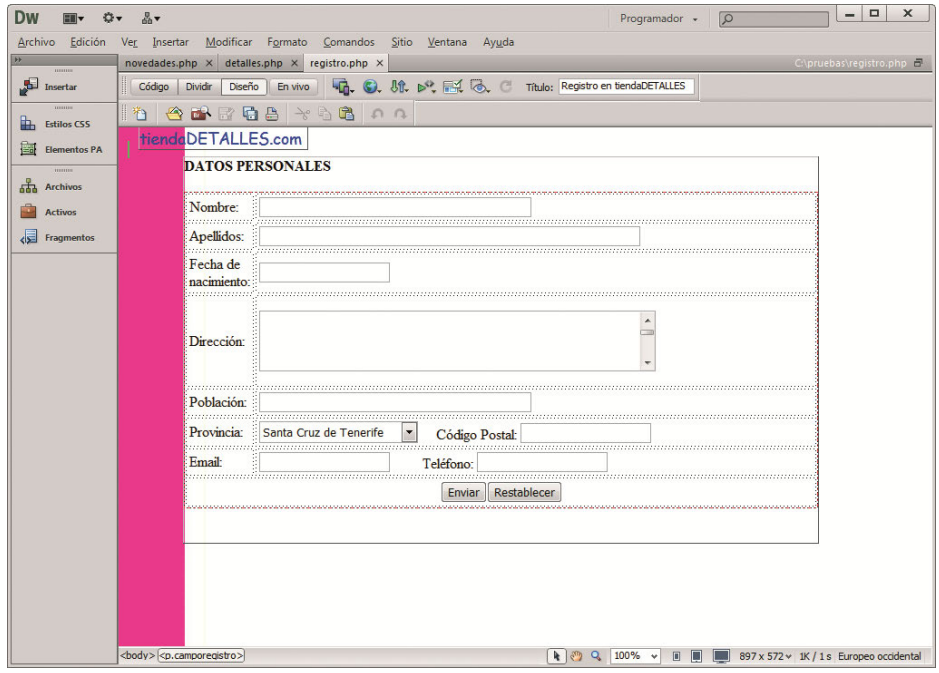
El método **Get** solo debe ser utilizado para transmitir poca información y que no tenga carácter privado. No es adecuado, por lo tanto, para transmitir el número de la tarjeta de crédito, los productos comprados, etc.

El método **Post** es el recomendable en prácticamente cualquier ocasión excepto en casos sencillos donde es aceptable transmitir la información en la propia URL. Si utilizamos el método Post, accederemos al array superglobal **$_POST**.

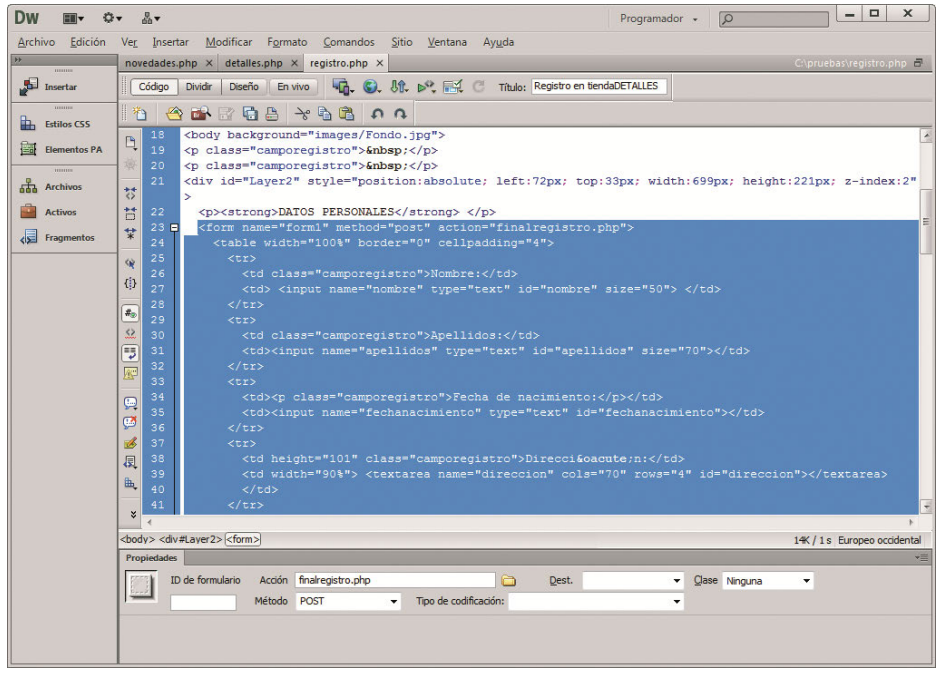
En la página siguiente vemos un ejemplo ilustrativo de lo que se conoce como un formulario HTML, donde el usuario introduce la información solicitada en distintos controles, como cuadros de texto, casillas de verificación, listas desplegables, etc.

Esa información es enviada al servidor web cuando pulsa en un botón del tipo **Enviar**.

En este caso se trata del formulario de registro en **tiendaDETALLES.com**. El usuario tiene que registrarse introduciendo los detalles necesarios aquí para poder comprar en la tienda virtual.



Debemos indicar dos características del formulario HTML: qué método utilizará para enviar la información al servidor web y en qué página se recogerá dicha información. Ambas están disponibles tanto en el código HTML como en el **Inspector de propiedades**, una vez seleccionado el formulario.



En la propiedad **Método** ya aparece que está seleccionado el envío mediante **Post**, aunque también podría utilizar el método Get, para que la información se pasara en la propia URL de la página.

Por otra parte, en la propiedad **Acción** indicaremos la página que recogerá esa información. Puede ser una página distinta o la misma en la que estás trabajando.

En este caso utilizaremos una página distinta para recoger y trabajar con la información enviada por el usuario.

Veamos cómo quedan representados estos dos detalles en el código de la página:

```
<form name="form1" method="post" action="finalregistro.php">
```

Fíjate que la etiqueta **<form>** tiene el atributo **method** establecido al valor **post**, indicando el método con el que se envía el formulario; y el atributo **action** establecido a la página que recogerá dicha información.

Finalmente, otro detalle importante es establecer el nombre de los controles del formulario, ya que después los utilizaremos para poder acceder a la información que allí haya introducido el usuario.

Lo importante es que sea un nombre lo suficientemente descriptivo para que después sea sencillo hacer referencia al control en el código php.

Veremos que utilizando el array **$_POST** será muy sencillo hacerlo porque se trata de un array asociativo en el que podremos utilizar el nombre de los controles que hemos establecido en el código HTML.

Utilizando Dreamweaver o un editor visual es fácil hacer todo esto, aunque, en definitiva, no es más que establecer al mismo valor los atributos **name** e **id** en la correspondiente etiqueta HTML. Por ejemplo:

```
<input name="email" type="text" id="email">
```

Hemos preparado la página para que el usuario pueda introducir los detalles necesarios para registrarse en la tienda virtual. En la próxima lección veremos cómo recoger dicha información.