Comunicación utilizando el envío de datos

Todas las aplicaciones tienen la necesidad en un momento determinado de comunicar distintas partes o módulos del desarrollo enviando datos entre ellos, como pueda ser el pasar el contenido de una variable de una página a un formulario o viceversa.

Cuando tengamos que compartir información entre las páginas de nuestra aplicación web, PHP nos ofrece distintos métodos que en función de las necesidades nos permiten la comunicación entre páginas utilizando el envío de datos.

En PHP podemos compartir información entre las distintas páginas del sitio web de las siguientes maneras:

- Utilizando los métodos GET y POST
- Utilizando Cookies
- Creando sesiones de trabajo *(esta forma de comunicación la veremos de forma más detallada en el próximo módulo del curso)*

Método GET

El paso de variables de una página a otra puede hacerse de varias formas, como ya hemos comentado.

El uso de GET en formularios, salvo que el desarrollador sepa muy bien lo que hace y los datos no sean comprometidos, no debe usarse para otras cosas diferentes a los formularios de búsqueda. El motivo es sencillo, el método GET lo que hace es pasar las variables y sus valores por la URL, es decir, no solo queda a la vista de cualquier usuario, sino que además la información puede quedar guardada en el historial del navegador.

Imaginar que se usa este método para identificar usuarios, y el nombre de usuario y la clave se quedan guardados en el historial del navegador de un ordenador, digamos de una biblioteca pública, el fallo en la seguridad sería catastrófico.

Uso de GET a través de la URL

Este método debido a como envía los datos por la URL, tiene una limitación en cuanto a la cantidad de información que se puede transmitir, fijada en 2048 caracteres.

Otra de las limitaciones del método GET es la imposibilidad de enviar archivos.

Si se utiliza con enlaces <a href=''> la sintaxis sería la siguiente:

```
Script_php?variable=valor&variable=valor&variable=valor…….
```

Se debe de poner un interrogante '?' a continuación del nombre de la página para indicar el comienzo del envío de datos, escribir el nombre de la variable seguida del signo '=' y el valor que le deseamos asignar. Si enviamos más de una variable las iremos separando con el carácter &.

```
<a href='http://pagina.php?id=100&categoria=industria&ref=AFD456X1'>Mostrar disponibilidad</a>
```

En el ejemplo anterior realizamos una llamada al script pagina.php, enviándole 3 variables con sus valores: id=100, categoría=industria y ref= AFD456X1.

La página que recibe la información, tendrá acceso a los mismos consultando el array global ***$_GET*** donde se han almacenado. Este array está formado por claves (nombre de la variable que se envía) y valores (valor de la variable).

Si quisiéramos consultar el contenido de las variables recibidas la forma sería:

```
$id_articulo=$_GET['id'];         // para la variable id
$cat_articulo=$_GET['categoria']; // para la variable categoria
$ref_articulo=$_GET['ref'];       // para la variable ref
```

Uso de GET a través de formularios HTML

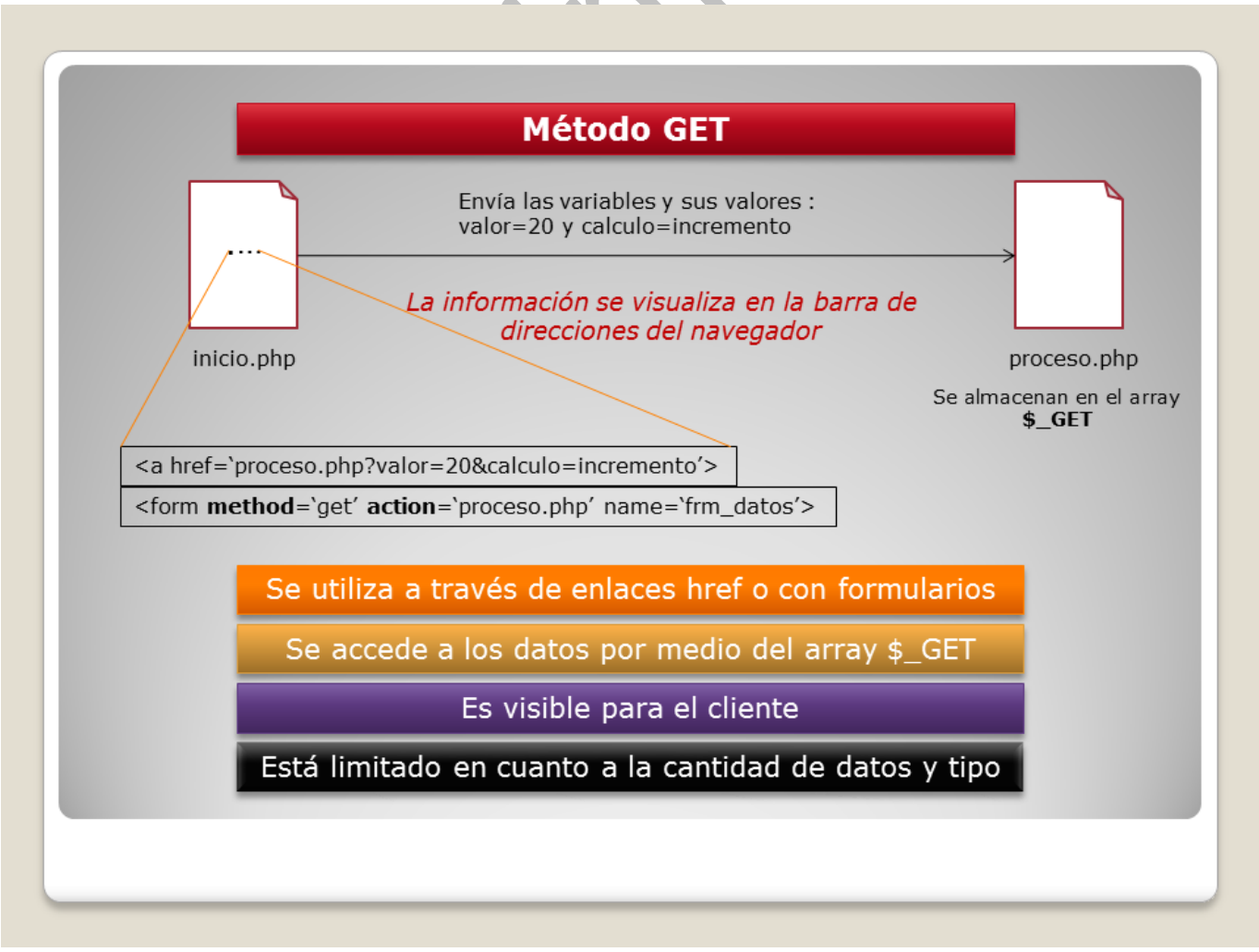
Si utilizamos el método GET por medio de formularios HTML no nos tendremos que preocupar de la sintaxis ya que el propio navegador se encargará de construir la URL.

Si se utiliza con formularios la sintaxis sería la siguiente:

```
<form method='get' action='pagina.php'>
```

Tendremos que indicar en el atributo 'method' del formulario la palabra get y en la acción 'action' el nombre de la página php a la que le deseamos enviar los datos.

La página que recibe la información, tendrá acceso a los mismos consultando el array global $_GET donde se han almacenado. Este array está formado por claves (nombre de la variable que se envía) y valores (valor de la variable).



Ejemplos

Envío de datos

Página HTML desde la que vamos a enviar los datos introducidos en el formulario a la página sumar.php.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Formulario GET</title>
</head>

<body>
<form method="GET" action="sumar.php">
    <p>Valor 1: <input type="text" name="T1" size="20"></p>
    <p>Valor 2: <input type="text" name="T2" size="20"></p>
    <p>Valor 3: <input type="text" name="T3" size="20"></p>
    <p><input type="submit" value="Sumar" name="B1"></p>
</form>
</body>

</html>
```

El mismo caso que el anterior, pero enviando los datos a través de la URL.

```
<p><a href="sumar.php?T1=10&T2=20&T3=30">Pulsar para enviar los datos de cálculo</a></p>
```

Recepción de datos

Página PHP que recibe los datos y los consulta en el array $_GET, realiza el cálculo y muestra el resultado.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Sumar GET</title>
</head>

<body>
<?php
$valor1 = $_GET['T1'];
$valor2 = $_GET['T2'];
$valor3 = $_GET['T3'];

$suma = $valor1 + $valor2 + $valor3;

echo "$valor1 + $valor2 + $valor3 = $suma";
?>
</body>
</html>
```

Método POST

El método POST nos permite también enviar información entre páginas como el método GET, pero sin sus limitaciones.

- La cantidad de información que podemos enviar mediante POST es bastante más grande (varios Mb) y vendrá limitada sólo por la propia configuración del PHP.
- El envío de datos no es visible, ni se almacena en la cache del navegador.
- Nos permite el envío de ficheros (*típico subir archivo*).

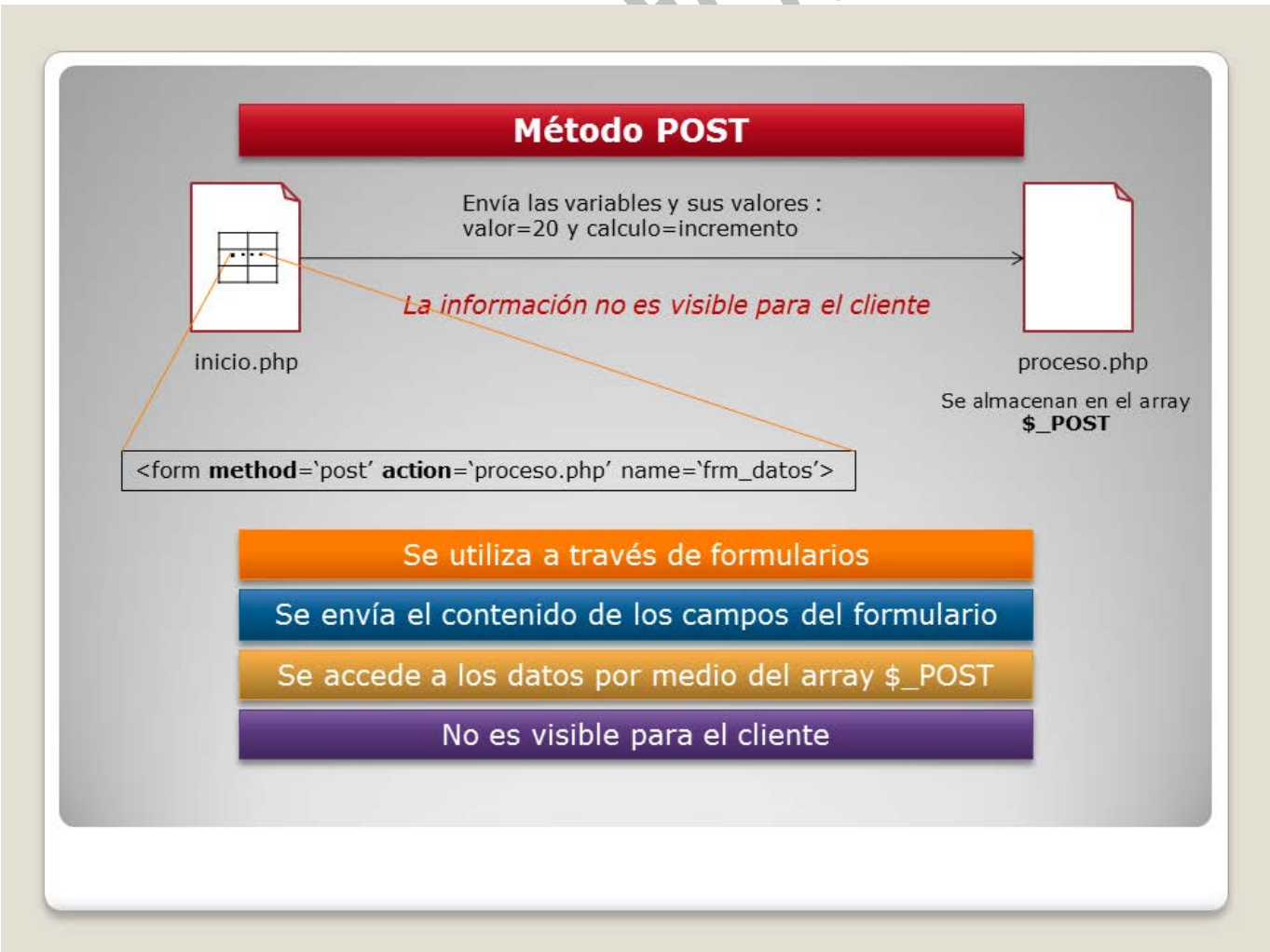
Se utiliza con formularios y la sintaxis es la siguiente:

```
<form method='post' action='pagina.php'>
```

- Tendremos que indicar en el atributo 'method' del formulario la palabra post y en la acción 'action' el nombre de la página php a la que le deseamos enviar los datos.
- Si se van a enviar archivos por medio de este método, tendremos que establecer el atributo ***enctype*** del formulario con el valor ***multipart/form-data***

```
<form method='post' action='pagina.php' enctype='multipart/form-data'>
```

La página que recibe la información, tendrá acceso a los mismos consultando el array global ***$_POST*** donde se han almacenado. Este array está formado por claves *(nombre del campo del formulario que se envía)* y valores *(valor del campo del formulario)*.



Ejemplos

Envío de datos

Página HTML desde la que vamos a enviar los datos introducidos en el formulario a la página sumar.php.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Formulario POST</title>
</head>

<body>
<form method="POST" action="sumar.php">
    <p>Valor 1: <input type="text" name="T1" size="20"></p>
    <p>Valor 2: <input type="text" name="T2" size="20"></p>
    <p>Valor 3: <input type="text" name="T3" size="20"></p>
    <p><input type="submit" value="Sumar" name="B1"></p>
</form>
</body>
</html>
```

Recepción de datos

Página PHP que recibe los datos y los consulta en el array $_POST, realiza el cálculo y muestra el resultado

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Sumar POST</title>
</head>

<body>
<?php
$valor1 = $_POST['T1'];
$valor2 = $_POST['T2'];
$valor3 = $_POST['T3'];

$suma = $valor1 + $valor2 + $valor3;

echo "$valor1 + $valor2 + $valor3 = $suma";
?>
</body>
</html>
```

Trabajo con Cookies

Las Cookies son un mecanismo que sirve para almacenar datos en el navegador del usuario remoto, para guardar variables de trabajo o simplemente para poder identificar al usuario cuando vuelva.

Para crearlas lo haremos usando la función ***setcookies***(). Las Cookies son parte de la cabecera HTTP, por tanto la función *setcookie*() debe ser llamada antes de que se produzca cualquier salida al navegador, es decir; tendremos que utilizarla antes de la etiqueta <html>.

Cualquier cookie enviada desde el cliente, automáticamente se convertirá en una variable PHP igual como ocurre con los métodos de datos GET y POST. Estas variables serán accesibles a través del array global ***$_COOKIE***.

Las Cookies tienen una caducidad, marcada por el tiempo de vida o de existencia en el navegador, por lo tanto las podremos tener por un intervalo de tiempo determinado o hasta que se cierre el navegador.

La sintaxis de la función para crear cookies es:

```
bool setcookie(string Cookie [, string Valor, int Caducidad, string Path, string Dominio, bool Seguridad]);
```

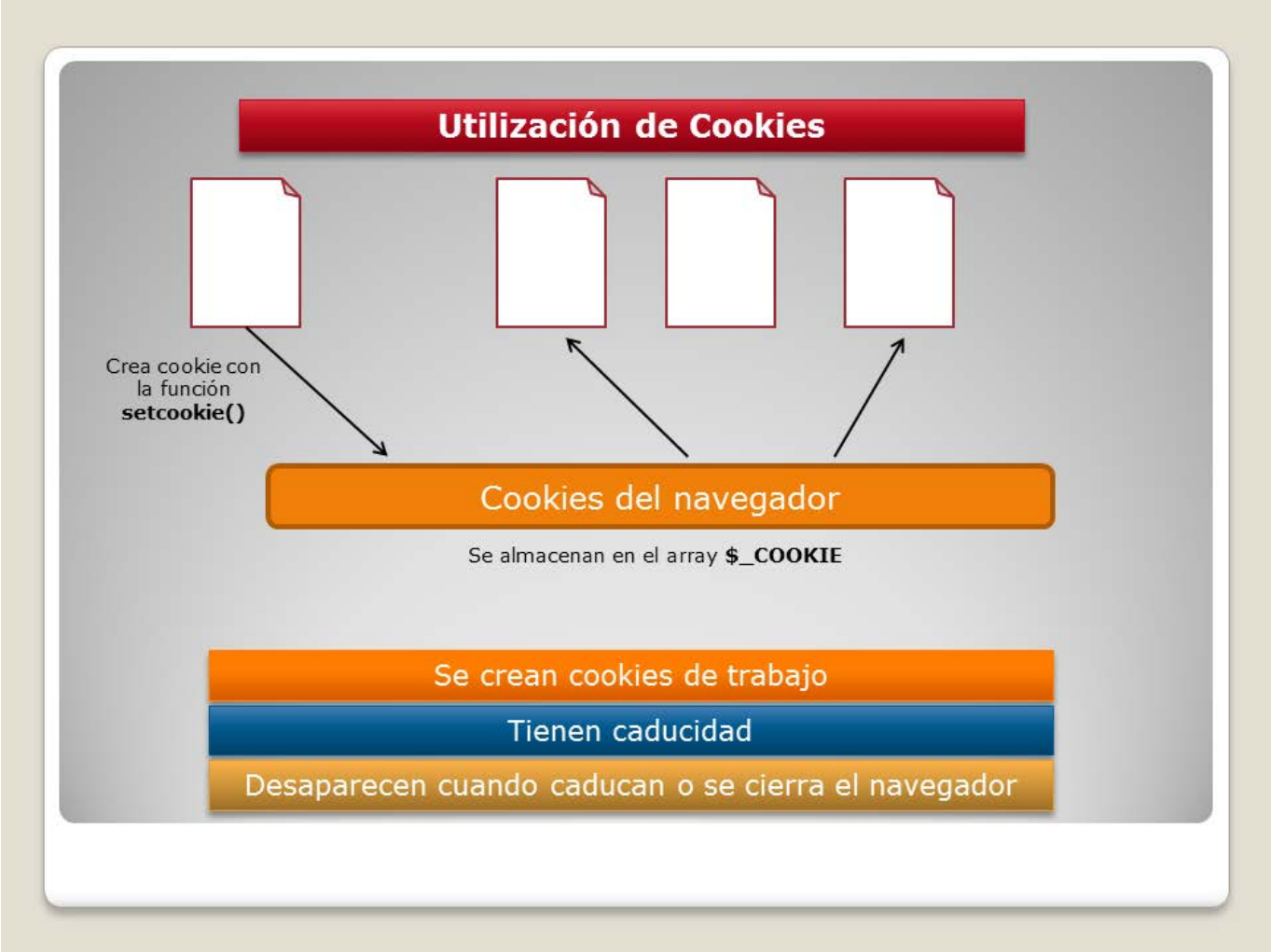
setcookie() define una cookie para ser enviada con el resto de la información de la cabecera. Las cookies deben enviarse antes de mandar cualquier otra cabecera *(esta es una restricción de las cookies, no de PHP)*. Esto requiere que situemos las llamadas a esta función antes de cualquier etiqueta <html> o <head>.

Todos los parámetros son opcionales excepto el primero que define el nombre de la cookie. Si sólo especificamos este parámetro, la cookie identificada por ese nombre será eliminada del cliente.

- **Cookie**: define el nombre de la cookie que se tiene que crear o eliminar
- **Valor**: valor que se le asigna. Enviaremos una cadena vacía “” para eliminar el valor de la cookie. Se accede al valor por medio del array global **$_COOKIE[‘*Cookie*’]**.
- **Caducidad**: define el tiempo de vida de la cookie en segundos a partir del tiempo actual. Para establecer su valor, utilizaremos la función time() más los segundos de vida: time()+60*10 define 10 minutos de vida para la cookie. Si se establece el valor 0, la cookie se destruirá cuando se cierre la sesión con el navegador, es decir; cuando cerremos la ventana.
- **Path**: La ruta dentro del servidor en la que la cookie estará disponible.
- **Dominio**: dominio para el cual la cookie está disponible en formato www.dominio.xxx ó dominio.xxx
- **Seguridad**: Indica que la cookie sólo debiera transmitirse por una conexión segura HTTPS desde el cliente.

Si queremos destruir una cookie antes de que finalice la sesión con el navegador, la volveremos a definir con un valor vacío y una duración anterior al momento actual.

```
setcookie("cookie", "", time()-42000, '/');
```



Ejemplos

Definir una cookie que tenga una vida de 20 días.

```php
<?php
/*
Si la cookie no existe la creamos asignándole el valor de la variable $usr y estableciendo una vida de 20 días
con el cálculo:
    HoraActual + Segundos_en_un_Minuto * Minutos_en_una_Hora * Horas_en_un_Dia * Dias
*/
if(! isset($_COOKIE['usuario'])) {
   setcookie('usuario', $usr, time()+60*60*24*20);
}
?>
```

Página PHP '*crea_cookies.php*' desde la que vamos a crear las cookies de trabajo para que el resto de las páginas del sitio web puedan acceder a las variables de trabajo.

```php
<?php
/*
Creamos 4 cookies de distintos tipos de datos y les asignamos una vida de 50seg a cada una.
Es muy importante tener en cuenta que la creación de cookies tiene que realizarse antes de
que se envíen las cabeceras al navegador, por lo que este código es obligatorio sea lo primero que escribamos
*/
setcookie('Numero','100',time()+50);
setcookie('Operacion','incremento',time()+50);
setcookie('FechaValidez','12/08/2013',time()+50);
setcookie('Frase','Estamos utilizando cookies de trabajo por parte del cliente',time()+50);
?>
<!doctype html>
<html>
<head>
<meta charset="utf-8">
```

```
<title>Crear Cookies</title>
</head>

<body>
<!– Creamos un enlace a la página proceso_cookie que nos mostrará el contenido de las cookies que hemos
creado y un contador por segundos -->
<p><a href="proceso_cookie.php">Proceso de Cookies</a></p>
</body>
</html>
```

Página '*proceso_cookie.php*' que muestra el contenido de las cookies creadas en el script anterior.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Muestra Cookies</title>
</head>

<body>
<?php
if(count($_COOKIE)==0) {
    echo "Las cookies de sesión han caducado.<br/>";
} else {
    echo "<p>El valor de las cookies de trabajo es:</p>";
/*
Utilizamos un bucle foreach para recorrer todos los elementos del array global $_COOKIE
separando los elementos por clave y valor y de esta manera después poder imprimir en la página
el nombre de la cookie '$clave' y su contenido '$valor'
*/
    foreach($_COOKIE as $clave => $valor) {
       echo "$clave -> $valor<br/>";
    }
}
?>
</body>
</html>
```