COOKIES Y SESIONES EN PHP

INTRODUCCIÓN

- El protocolo que se usa para navegar por la web es HTTP o HTTPS.
- Básicamente un cliente (navegador) hace una petición (request) y obtiene una respuesta (response) del servidor.
- El problema está en que HTTP es un protocolo sin estado, es decir no se guarda el estado de la transacción.
- Esto para muchas de las aplicaciones actuales es un problema, ya que en muchas se nos pide, por ejemplo, una identificación que nos permita movernos por ella sin tener que estar identificándonos en cada una de las páginas de la aplicación.
- Debemos, por tanto, poder recordar esa identificación y para ello, necesitamos poder guardar de alguna manera el estado de la comunicación y poder conocerlo en las diferentes páginas de la aplicación Web.

INTRODUCCIÓN

Hay varias formas de generar/guardar un estado.

- **USO DE LA IP**
  - Una opción podría ser usar como método de autentificación la dirección IP del cliente, a la que tenemos acceso mediante la variable superglobal $_SERVER[“REMOTE_ADDR”]
  - Sin embargo, no es un método seguro, ya que el usuario puede acceder desde servidores proxy o utilizando otras tecnologías que no permitan identificar con seguridad su dirección IP.

INTRODUCCIÓN

Hay varias formas de generar/guardar un estado.

- **PASO DE PARÁMETROS MEDIANTE CADENA DE CONSULTA**
  - Podemos pasar parámetros que indiquen el estado a través de la URL con el método GET.
  - Por ejemplo si en nuestro navegador escribimos https://www.google.es/#q=PHP se abre la página de Google en España y se muestran las páginas que según Google se relacionan con PHP.
  - Esta opción no es segura, ya que la información que se envía es susceptible de ser vista y capturada por terceros.

INTRODUCCIÓN

Hay varias formas de generar/guardar un estado.

- **PASO DE PARÁMETROS MEDIANTE EL MÉTODO POST**
  - El método POST también permite pasar información entre páginas. En este caso los datos no viajan en la URL del mensaje sino en el cuerpo del mensaje HTTP, pero la información se puede ver en el navegador.
  - Muchas veces se usan controles de formulario ocultos tipo **“hidden”** para pasar información adicional sin que el usuario la perciba en el formulario. Cuando los datos se envían, se estarán enviando tanto los datos visibles del formulario, como los ocultos, aunque esta información es visible en el código fuente de la página.

INTRODUCCIÓN

Hay varias formas de generar/guardar un estado.

- **COOKIES**
  - Ficheros que se guardan en el entorno del navegador del usuario. El mismo programa, abierto en dos navegadores distintos tiene diferentes cookies.
- **SESIONES**
  - Con el término sesión, nos referimos a una forma de guardar datos de manera similar a las cookies, pero en el servidor.

COOKIES

COOKIES

- Una cookie es un fichero de texto que un sitio web guarda en el entorno del navegador del usuario.
- El navegador las crea a petición del navegador. A partir de ese momento el servidor puede leer y escribir contenido en la cookie creada.
- Su uso más típico es el almacenamiento de las preferencias del usuario (por ejemplo, el idioma en que se deben mostrar las páginas), para que no tenga que volver a indicarlas la próxima vez que visite el sitio.

COOKIES

Restricciones de seguridad:

- Los servidores web solo pueden acceder a cookies establecidas a su propio dominio. Este dominio es establecido por el navegador cuando el servidor crea una nueva cookie y solo puede ser un dominio o subdominio del servidor.
- Las cookies no pueden ser más grandes de 4096 bytes.
- El número máximo de cookies por dominio que admite el navegador es 20.
- Un navegador puede almacenar hasta 300 cookies en total.

COOKIES

- Las cookies se pasan en la cabecera del mensaje HTTP.
- Es decir, desde el momento en que desde una página Web se guardan datos mediante cookies, todas nuestras peticiones y respuestas llevan anexas los datos de las cookies que hemos almacenado, y podremos usarlos en cualquier momento.
- Por tanto, cuando se invoque a la función **setcookie()**, esta ha de ser llamada antes de cualquier etiqueta HTML o de salida en pantalla, si no se producirá un error (igual que con la función header).

COOKIES

- En PHP, para almacenar una cookie en el navegador del usuario, se utiliza la función **setcookie()** con el nombre de la cookie que quieres guardar como único parámetro obligatorio.
- **INCONVENIENTE:** el usuario puede bloquear o borrar las cookies, por lo que no tenemos control sobre estas acciones.
- Su uso generó polémica ya que se estaba guardando información en el ordenador del cliente sin su conocimiento/autorización. Actualmente hay que avisar de que estamos utilizando cookies.

COOKIES

- La sintaxis es:

setcookie (nombre [,valor [,expiración [,ruta [,dominio [,segura [solohttp]]]]]])

- **Nombre:** es el nombre que le damos a la cookie. Lo usaremos para poder consultar su valor.
- **Valor:** valor que le damos a la cookie.
- **Expiración:** por defecto toma el valor 0 que significa que la cookie caduca cuando se cierre el navegador con la que se creó. Otra posibilidad es indicarle una fecha en formato Unix. Por ejemplo, time()+600 indica que la cookie expira en 10 minutos.
- **Dominio:** si no se indica toma como dominio en el que se aplica la cookie el dominio actual, pero se puede indicar otro.
- **Segura:** por defecto es false. Si indicamos true, indica que la cookie solo se almacena si estamos comunicándonos mediante protocolo seguro.
- **soloHttp:** valor booleano que indica que la cookie solo está disponible mediante el protocolo http y no, por ejemplo desde JS con idea de intentar disminuir riesgos de un ataque realizado con JS.
- **Nosotros vamos a trabajar con las tres primeras opciones.**

COOKIES

- **Las cookies tiene fecha de caducidad, en caso de no definirla, la cookie se eliminará al cerrar el navegador.**
- Para recuperar la información almacenada en una cookie se utiliza la variable superglobal **$_COOKIE**, que contiene un array asociativo con las cookies del dominio.

COOKIES

- **Para eliminar una cookie, usaremos la función setcookie pasándole una fecha anterior a la actual.**

COOKIES

- **Ejemplo: contador de visitas y fecha y hora en la que se produjo**

```php
<?php

    if(isset($_COOKIE['visitas'])){
        $_COOKIE['visitas'] ++;
        setcookie("visitas",$_COOKIE['visitas']);
        setcookie("hora",date("d-m-Y H:i:s"));
        echo "esta es tu visita número: {$_COOKIE['visitas']}<br/>";
        echo "has visitado esta página: {$_COOKIE['hora']}";
    }
    else{
        setcookie('visitas',1);
        setcookie("hora",date("d-m-Y H:i:s"));
    /* la primera vez si se intenta mostrar la cookie da error porque
     * las cookies se actualizan al volver a refrescar la pág*/
        echo "es la primera vez que visitas la página";
    }
```

COOKIES

- Ejemplo: preferencias de usuario
- El usuario debe seleccionar en un formulario sus preferencias en cuanto a color de fondo, color de letra, y tipografía (entre tres opciones). Además debe introducir el nombre y los apellidos.
- Al enviar el formulario se redirige a una página en la que se le saluda por sus nombre y que está configurada con los colores y tipo de letra elegido. Las opciones seleccionadas se guardarán en cookies, de forma que si se vuelve a ejecutar el script principal, si se detecta que hay cookies, se redirige automáticamente a esta segunda página, configurada con las selecciones realizadas.
- La segunda página contiene un enlace que al pulsarlo redirige a la primera página, borrando además las cookies con las opciones elegidas.

Ejemplo: preferencias de usuario Practica1-index.php

```php
<?php
include "practical-comprobar.php";
if(hayCookie()){
    /*si tenemos preferencias ya guardadas, directamente vamos a saludo.php
     * Si no hemos completado todos los datos del formulario no tendrá el
     * valor guardado de todas las cookies, y por tanto la función hayCookie
     * devolverá false.
     */
    header("location:practical-saludo.php");
}
?>
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Formulario de recogida de preferencias</title>
</head>
<body>
<form action="practical-saludo.php">
    <fieldset>
        <legend>Escoge tus preferencias</legend>
        <label for="nombre">Nombre</label>
        <input type="text" name="nombre" id="nombre"/><br/><br/>
        <label for="apellidos">Apellidos</label>
        <input type="text" name="apellidos" id="apellidos"/><br/><br/>
        <label for="fondo">Color de fondo</label>
        <input type="color" name="fondo" id="fondo" value="#FFFFFF"/><br/><br/>
        <label for="frente">Color de letra</label>
        <input type="color" name="frente" id="frente"/><br/><br/>
        <label for="letra">Tipo de letra</label>
        <select name="letra" id="letra">
            <option value="'Shadows Into Light', cursive">
                Shadows Into Light
            </option>
            <option value="'Slabo 27px', serif">Slabo 27px</option>
            <option value="'Roboto', sans-serif">Roboto</option>
        </select><br/><br/>
        <button name="enviar">Enviar</button>
    </fieldset>
</form>
</body>
</html>
```

Ejemplo: preferencias de usuario Practica1-saludo.php (I)

```php
<?php
    include "practical-comprobar.php";
    include "practical-guardar.php";
    //nos dice si tenemos preferencias de usuario
    $hayPreferencias=true;
    $aConf=null;
    /*
     * Si no hay cookies mira si hay datos del formulario. Si no hay datos
     * pone la variable preferencias a false. Si los hay los guarda en un array.
     *
     * Si hay datos de cookies los guarda en un array
     */
    if(hayCookie()==false){
        if(hayDatos()==false){
            $hayPreferencias=false;
        }
        else{
           $aConf = guardarDatos($_REQUEST);
        }
    }
    else{
        $aConf = guardarDatos($_COOKIE);
    }
    if($hayPreferencias==false)
        //si no tenemos preferencias, volvemos al formulario
        header("location:practical-index.php");
    else{
        //grabar cookies con las preferencias de usuario
        setcookie("nombre",$aConf["nombre"],time()+60*60*24);
        setcookie("apellidos",$aConf["apellidos"],time()+60*60*24);
        setcookie("fondo",$aConf["fondo"],time()+60*60*24);
        setcookie("frente",$aConf["frente"],time()+60*60*24);
        setcookie("letra",$aConf["letra"],time()+60*60*24);
    }
?>
```

COOKIES

Ejemplo: preferencias de usuario Practica1-saludo.php (II)

```php
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Saludo</title>
    <link href='http://fonts.googleapis.com/css?family=Shadows+Into+Light|Slabo+27px|Roboto' rel='stylesheet' type='text/css'>
    <style>
        body{
            background-color:<?=$aConf["fondo"];?>;
            color:<?=$aConf["frente"];?>;
            font-family:<?=$aConf["letra"];?>;
        }
        a{
            background-color:<?=$aConf["fondo"];?>;
            color:<?=$aConf["frente"];?>;
        }
    </style>
</head>
<body>
<h1>Hola <?=$aConf["nombre"]." ".$aConf["apellidos"];?></h1>

<p><a href="practical-borrar.php">Borrar cookies y volver a cambiar las preferencias</a></p>
</body>
</html>
```

COOKIES

Ejemplo: preferencias de usuario Practica1-comprobar.php

```php
<?php
/*
 * Devuelve true si existen todas las cookies.
 */
function hayCookie(){
    return isset($_COOKIE["nombre"]) && isset($_COOKIE["apellidos"])
        && isset($_COOKIE["fondo"]) && isset($_COOKIE["frente"])
        && isset($_COOKIE["letra"]);
}
/*
 * Devuelve true si han llegado todos los datos.
 */
function hayDatos(){
    return isset($_REQUEST["nombre"]) && isset($_REQUEST["apellidos"])
        && isset($_REQUEST["fondo"]) && isset($_REQUEST["frente"])
        && isset($_REQUEST["letra"]);
}

?>
```

COOKIES

Ejemplo: preferencias de usuario Practica1-borrar.php

```php
k?php
    //borra las cookies y nos devuelve al formulario
    setcookie("nombre"," ",time()-6000);
    setcookie("apellidos"," ",time()-6000);
    setcookie("fondo"," ",time()-6000);
    setcookie("frente"," ",time()-6000);
    setcookie("letra"," ",time()-6000);
    header("location:practical-index.php");
?>
```

COOKIES

Ejemplo: preferencias de usuario Practica1-guardar.php

```php
<?php
/*
 * Guarda en un array los datos que nos ha llegado por parámetro (que puede ser
 * el contenido de $_REQUEST o el de $_COOKIE y lo devuelve.
 */
function guardarDatos($array){
    $aDatos["nombre"] =$array["nombre"];
    $aDatos["apellidos"] =$array["apellidos"];
    $aDatos["fondo"] =$array["fondo"];
    $aDatos["frente"] =$array["frente"];
    $aDatos["letra"] =$array["letra"];
    return $aDatos;
}
?>
```

SESIONES

https://www.php.net/manual/es/book.session.php

SESIONES

- Entendemos por sesión la actividad que realiza un usuario en su navegador desde que lo abre hasta que lo cierra.
- Sin embargo, en programación, y en nuestro caso en PHP, cuando hablamos de sesiones nos referimos a una forma de guardar datos de manera similar a las cookies, pero en el servidor.
- Así, evitamos los inconvenientes derivados de almacenar información en el ordenador del usuario.

SESIONES

- Cuando se crea una sesión, se le asigna un identificador aleatorio único (SID), de forma que cada sesión se identifica de forma única.
- Asociado a este identificador de sesión, se crea un archivo en el servidor en el que se almacenarán físicamente los datos de la sesión, por lo que ya no dependemos del usuario para poder tener accesible esa información.

SESIONES

- Las sesiones guardan información relativa a un usuario concreto. Esta información puede ser desde su nombre hasta el número de productos que ha depositado en una cesta de la compra de una tienda online.
- Cada usuario distinto de una misma página web tiene su propio (SID).

SESIONES

- Existen dos maneras de mantener el SID entre páginas de un sitio WEB:
  - **Usando una cookie que almacena el valor que identifica al usuario (SID) en el servidor web**. En el servidor web están almacenados todos los datos de la sesión y se accede a ellos cada vez que se pasa de página gracias al identificador almacenado en la cookie **(es la opción por defecto y la que vamos a usar)**.

    Si las cookies se borran manualmente, las sesiones abiertas se cierran
  - **Pasando el SID como parámetro en la URL**. Esta opción no es recomendable, ya que no es conveniente pasar esa información de forma visible en la URL.

SESIONES

OBTENER DATOS DE SESIÓN: Hay dos funciones con las que obtener (y cambiar) los datos de la sesión:

- **session_id():**
  - Si la llamamos sin parámetros devuelve el identificador de sesión actual.
  - Si le pasamos nuestro propio identificador de sesión, este se convierte en el identificador de la sesión actual. Esto es peligroso si no disponemos de un buen generador de identificadores ya que corremos el riesgo de tener dos sesiones con el mismo identificador, de forma que no podremos distinguir una de otra, y podríamos provocar que se vean datos de una en otra (no recomendable).
- **session_name():**
  - Si la invocamos sin parámetros, devuelve el nombre de sesión actual que por defecto se llama **PHPSESSID.**
  - Si le pasamos como parámetro un nombre de sesión podemos dificultar que la sesión sea identificada por terceros, ya que el nombre PHPSESSID es conocido por todos los programadores de PHP, lo que facilita su identificación para la captura y robo de la sesión.

SESIONES

- Ejemplo: en el siguiente ejemplo mostramos por pantalla el identificador de sesión que se nos ha asignado, el nombre de la sesión (que también es el nombre de la cookie que se emplea para guardar el identificador de la sesión, y por último se muestran los parámetros de la cookie que almacena el identificador de la sesión.

```
<?php
  session_start();
  echo 'session_id(): '.session_id();
  echo "<br />\n";
  echo 'session_name(): '.session_name();
  echo "<br />\n";
  print_r(session_get_cookie_params());
?>
```

SESIONES

```php
<?php
  session_start();
  echo 'session_id(): '.session_id();
  echo "<br />\n";
  echo 'session_name(): '.session_name();
  echo "<br />\n";
  print_r(session_get_cookie_params());
?>
```

- Salida en pantalla:

```
session_id(): mnm7ck1qffh7o9bcsnk48as8j8
session_name(): PHPSESSID
Array ( [lifetime] => 0 [path] => / [domain] => [secure] => [httponly] => [samesite] => )
```

SESIONES

- El manejo de sesiones en PHP está bastante automatizado.
- El SID viaja entre el cliente y el servidor, bien como parte de la URL o en un encabezado HTTP si se guarda en una cookie.
- En ambos casos, esto plantea un posible problema de seguridad. El SID puede ser conseguido por otra persona, y a partir del mismo obtener la información de la sesión del usuario.
- La manera más segura de utilizar sesiones, es almacenando los SID en cookies y utilizar HTTPS para cifrar la información que se transmite entre el servidor web y el cliente.

SESIONES

CONFIGURACIÓN

- Algunas directivas del fichero php.ini relativas a las sesiones son:

SESIONES

DIRECTIVA	SIGNIFICADO
session.use.cookies	1 = el SID se almacena en cookies. 0 = el SID se almacena en la URL
session.use_only_cookies	se debe poner un 1 cuando se usan las cookies para guardar el SID, y no quieres que se reconozcan Los SID que se puedan pasar como parte de la URL.
session.save_handler	Indica a PHP cómo guardar los datos de la sesión del usuario. Hay 4 opciones: ficheros que es el valor por defecto y funcionará sin problemas en la mayoría de los casos (files), memoria (mm), base de datos SQLite (SQL) o usando funciones definidas por el programador (user).
session.name	Nombre de la cookie que se utilizará para guardar el SID. Su valor por defecto es PHPSESSID
session.auto_start	Su valor por defecto es 0, en cuyo caso se debe usar la función session_start para gestionar el inicio de las sesiones. Si cambiamos su valor a 1 PHP, activa de forma automática el manejo de sesiones.
session.cookie_lifetime	Si utilizas la URL para propagar el SID, este se perderá cuando cierres tu navegador. Si utilizas cookies, el SID se mantendrá mientras no se destruya la cookie. En su valor por defecto (0), las cookies se destruyen cuando se cierra el navegador. Si quieres que se mantenga el SID durante más tiempo, debes indicar en esta directiva ese tiempo en segundos.
session.gc_maxlifetime	Indica el tiempo en segundos que se debe mantener activa la sesión, aunque no haya ninguna actividad por parte del usuario. Su valor por defecto es 1440 (24 minutos) pasado el cual sin que haya habido actividad en esos 24 minutos por parte del usuario, se cierra su sesión automáticamente.

SESIONES

INICIO DE UNA SESIÓN:

- Debemos utilizar la función **session_start()** para **crear una nueva sesión o reanudar la existente**.
- Esta función devuelve false en caso de no poder iniciar o restaurar la sesión y true si la ha iniciado correctamente.

SESIONES

USO DE LA SESIÓN PREVIAMENTE CREADA:

- Si la sesión ya existe, se sigue utilizando la misma y mediante el array asociativo **$_SESSION** podemos acceder a los datos que tuviera guardados.

SESIONES

- Como crear/iniciar sesión requiere utilizar cookies, y éstas se transmiten en los encabezados HTTP, debes tener en cuenta que para poder iniciar una sesión utilizando **session_start()**, tendrás que hacer las llamadas a esta función antes de que la página web muestre información en el navegador.
- Además, **todas las páginas que necesiten utilizar la información almacenada en la sesión, deberán ejecutar la función session_start().**
- Mientras la sesión permanece abierta, puedes utilizar la variable superglobal **$_SESSION** para añadir o acceder a la información de la sesión.
- Las sesiones caducan cuando el navegador se cierra, esto es lo más recomendable, ya que si hacemos que dure más estamos haciendo que nuestra aplicación sea más insegura.

SESIONES

USAR VARIABLES DE SESIÓN

- Una variable de sesión permite asociar una clave a un valor de la sesión actual usando la siguiente sintaxis:
  $_SESSION[“nombre”]=valor.
- Para obtener el valor almacenado se usando la clave o nombre que le asignamos:

```
$variable = $_SESSION[“nombre”];
```

SESIONES

BORRAR DATOS DE SESIÓN:

- Funciones para eliminar la información almacenada en la sesión:
  - **session_unset():** Elimina **todas** las variables almacenadas en la sesión actual, pero no elimina la información de la sesión del dispositivo de almacenamiento usado.
  - **unset ($_SESSION[‘nombre’]):** elimina una variable de sesión concreta.
  - **session_destroy():** Elimina completamente la información de la sesión del dispositivo de almacenamiento.

SESIONES

- Ejemplo: creamos una página que crea/inicia sesión y guardamos en ella un elemento llamado “nombre” con valor “Jorge”

```
<?php
    session_start();
    $_SESSION[“nombre”]=”Jorge”;
?>

<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <p>Se ha iniciado la sesión</p>
        <a href="pagina2.php">Ir a página </a>
    </body>
</html>
```

En este código se crea una nueva sesión y se asigna a ella el valor Jorge asociado al identificador “nombre”. Además se muestra el mensaje “Se ha iniciado sesión” y un enlace a la página pagina2.php.

SESIONES

- Ejemplo: el código de pagina2.php sería:

```
<?php
    session_start();
?>
<!DOCTYPE html>
 <html>
    <head>
       <meta charset="UTF-8">
    </head>
    <body>
        <?="Hola ".$_SESSION["nombre"];?>
    </body>
</html>
```

Se muestra el texto “Hola Jorge”, resultado de acceder a los datos de la sesión correctamente.

SESIONES

- Por ejemplo, para contar el número de veces que el usuario visita la página, puedes hacer:

```
<?php
  //Iniciamos la sesión o recuperamos la anterior sesión existente
  session_start();
  // Comprobamos si la variable ya existe
  if (isset($_SESSION['visitas']))
      $_SESSION['visitas']++;
  else
      $_SESSION['visitas'] = 1;
?>
```

SESIONES

- Si en lugar del número de visitas, quisieras almacenar el instante en que se produce cada una de ellas, la variable de sesión "visitas" deberá ser un array, y por tanto tendrás que cambiar el código anterior por:

```
<?php
  //Iniciamos la sesión o recuperamos la anterior sesión existente
  session_start();
  //En cada visita añadimos un valor al array "visitas"
  $_SESSION['hora'][] = mktime();
?>
```

SESIONES

- La función **session_status()** devuelve el estado de la sesión actual:
  - **PHP_SESSION_DISABLED** si las sesiones están deshabilitadas.
  - **PHP_SESSION_NONE** si las sesiones están habilitadas, pero no existe ninguna.
  - **PHP_SESSION_ACTIVE** si las sesiones están habilitadas, y existe una.
- **Su uso es muy recomendable** ya que evitamos los errores que se puedan producir al intentar iniciar una sesión si ya está iniciada (suele pasar al incluir un script mediante include o require, que intente crear/recuperar una sesión ya creada o inicializada previamente).
- Ejemplo

```
if(session_status() == PHP_SESSION_NONE)
        session_start();
```

SESIONES

- Trabajando con sesiones se nos puede plantear el siguiente problema:
- Imaginemos que tenemos una aplicación web con tres páginas. En la primera hay un formulario de login en el que simplemente se pide al usuario que introduzca su nombre, para en la segunda página mostrar los datos de la sesión, y en la tercera borrarlos.
- ¿Qué crees que ocurrirá si una vez logueado un usuario intenta acceder a la página 2 o 3 directamente (le puede salir en el historial), sin pasar por el login?

SESIONES

Observa este archivo.
Se encarga de controlar el flujo del programa, incluyendo uno u otro archivo en función de lo que se haya pulsado en las demás páginas de la aplicación.

Script index.php

```php
session_start();
if(!empty($_REQUEST['identificar'])){
    if(isset($_POST['nombre'])&& !empty($_POST['nombre'])){
        $_SESSION['usuario'] = $_POST['nombre'];
        include_once ('pagina2.php');
        die;
    }
}
if (isset($_POST['borrar']) && isset($_SESSION['usuario'])){
    //si existe usuario y ha pulsado borrar se va a otra página
    include_once ('pagina3.php');
    die;
}
  else if(isset($_POST['borrar']) || empty($_REQUEST))
else
{
    include_once ('login.php');
    die;
}
```

SESIONES

Script login.php

Este archivo contiene el formulario de login.
Cuando el usuario pulsa el botón enviar se ejecuta el archivo index.php que hemos visto antes.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <!-- formulario  que pide el nombre. Al el botón de envío se redirige al archivo index.php-->
        <form action="index.php" method="post">
            <label>Introduce tu nombre</label>
            <input type="text" name="nombre" autofocus="autofocus">
            <input type="submit" name ="identificar" value="enviar">
        </form>
    </body>
</html>
```

SESIONES

```php
    if(session_status() == PHP_SESSION_NONE)
        session_start();
    if(isset($_SESSION['usuario']))
    {
            $nombre = $_SESSION['usuario'];
            echo "<br/>Página2.php - USUARIO logeado<br/>";
            echo "<br/>Página2.php - usuario = ".$_SESSION['usuario']."<br/>";
            echo "<br> pagina2.php - ID sesión= ".session_id()."<br/>";
    }
    else{
        echo "<br> pagina2.php - No SE HA REGISTRADO O NO HAY SESIÓN, debería redirigir a la a la página de login </br>";
        include('login.php');
        die;
    }
?>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <form action="index.php" method="post">
            <p><?php if(isset($nombre)) echo "Hola ".$nombre ?></p>
            <label>Si pulsas el botón va a la página 3 donde se borran la sesion</label>
            <input type="submit" name = "borrar" value="Borrar sesión">
        </form>
    </body>
</html>
```

El archivo pagina2.php comprueba si existe el usuario y lo muestra. En caso contrario muestra un mensaje de error ye incluye el fichero de login. Al pulsar el botón del formulario pedirá al servidor la página index.php llegándole que ha pulsado el botón borrar.

SESIONES

Script pagina3.php

Este archivo contiene comprueba si existe la sesión, si no existe la inicia/crea
Borra los datos de la sesión y vuelve a la página index, que la redirigirá a login.

```php
    if(session_status() == PHP_SESSION_NONE)
        session_start();
    if(isset($_SESSION['usuario'])){
        $nombre = $_SESSION['usuario'];
        session_unset();
        session_destroy();

    }
    else{
        echo "<br> pagina3.php - No SE HA REGISTRADO O NO HAY SESIÓN, debería redirigir a la a la página de
        login </br>";
    }
?>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <form action="index.php" method="post">
            <p> <?php if(isset($nombre)) echo "Hola ".$nombre ?></p>
            <label>Si pulsas el botón va a la página login</label>
            <input type="submit" name="loguear"  value="volver">
        </form>
    </body>
</html>
```

SESIONES

- La aplicación no debería dejarle acceder a esa página directamente, pero tenemos datos de sesión guardados que indican que se ha logueado en algún momento y por tanto puede acceder.
- ¿Qué solución propones?

SESIONES

- Hay varias soluciones posibles, tantas como puedas plantearte, desde controlar el tiempo que lleva inactivo y borrar manualmente la sesión, hasta …. lo que imagines.
- Una posible solución muy sencilla de implementar es crear una variable de sesión o una cookie que se encargue de controlar si ha pasado por el proceso de login o no.
- En las páginas 2 y 3 se comprueba que esa variable existe, y de ser así se borra para evitar que pueda acceder directamente a la página.

SESIONES

Script index.php

Cuando un usuario se identifica, creamos una variable de sesión con valor true que nos servirá para comprobar si el usuario de ha logueado.

```php
session_start();
if(!empty($_REQUEST['identificar'])){
    if(isset($_POST['nombre'])&& !empty($_POST['nombre'])){
        $_SESSION['usuario'] = $_POST['nombre'];
    /* creamos una variable se sesión en el caso de que
     * haya habido una identificación positiva. Esa variable va a tener valor
     * true.
     */
        $_SESSION['login'] = "true";
        include_once ('pagina2.php');
        die;
    }
}
if (isset($_POST['borrar']) && isset($_SESSION['usuario'])){
    // si existe usuario y se va a otra página e indica que hay login
    // el botón borrar está en la página 2, y debería haber usuario logueado.
    $_SESSION['login']="true";
    include_once ('pagina3.php');
    die;
}
//else if(isset($_POST['borrar']) || empty($_REQUEST))
else
{
    //en cualquier otro caso redirige a login
    include_once ('login.php');
    die;
}
```

SESIONES

Script login.php

Este archivo no sufre ningún cambio.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <!-- formulario  que pide el nombre. Al el botón de envío se redirige al archivo index.php-->
        <form action="index.php" method="post">
            <label>Introduce tu nombre</label>
            <input type="text" name="nombre" autofocus="autofocus">
            <input type="submit" name ="identificar" value="enviar">
        </form>
    </body>
</html>
```

```php
    if(session_status() == PHP_SESSION_NONE)
        session_start();

    if(isset($_SESSION['login']) && $_SESSION['login'] == "true")
    {
        if(isset($_SESSION['usuario']))
        {
            $nombre = $_SESSION['usuario'];
            echo "<br/>Página2.php - USUARIO logeado<br/>";
            echo "<br/>Página2.php - usuario = ".$_SESSION['usuario']."<br/>";
            echo "<br> pagina2.php - ID sesión= ".session_id()."<br/>";
        /* por si acaso la borra. Si tuviera que ir a otra página y el usuario es
         * necesario, volveríamos a crearla
         */
            unset($_SESSION['login']);
        }
    }
    else{
        echo "<br> pagina2.php - No SE HA REGISTRADO O NO HAY SESIÓN, debería redirigir a la a la página de
        login </br>";
        include('login.php');
        die;
    }
?>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <form action="index.php" method="post">
            <p><?php if(isset($nombre)) echo "Hola ".$nombre ?></p>
            <label>Si pulsas el botón va a la página 3 donde se borran la sesion</label>
            <input type="submit" name = "borrar" value="Borrar sesión">
        </form>
    </body>
</html>
```

El archivo pagina2.php comprueba si existe la variable de sesión con la que comprobamos si un usuario se ha logueado. Si es así, muestra los datos y borra la variable de sesión de login.
Si no, redirige a login.php

SESIONES

```php
    if(session_status() == PHP_SESSION_NONE)
        session_start();

    if(isset($_SESSION['login']))
    {
        if(isset($_SESSION['usuario']))
            $nombre = $_SESSION['usuario'];
        session_unset();
        session_destroy();
    }
    else{
     /*
      * Si no existe la variable de sesión de login ha llegado a esta página
      * sin identificarse primero.
      */
        echo "<br> pagina3.php - No SE HA REGISTRADO O NO HAY SESIÓN, debería redirigir a la a la página de
        login </br>";
    }
'>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <form action="index.php" method="post">
            <p> <?php if(isset($nombre)) echo "Hola ".$nombre ?></p>
            <label>Si pulsas el botón va a la página login</label>
            <input type="submit" name="loguear" value="volver">
        </form>
    </body>
</html>
```

El script pagina3.php comprueba si existe la variable de login con la que comprobamos que un usuario se ha logueado.
El resto es igual