Sistema de autentificación

PHP. Múltiples usuarios

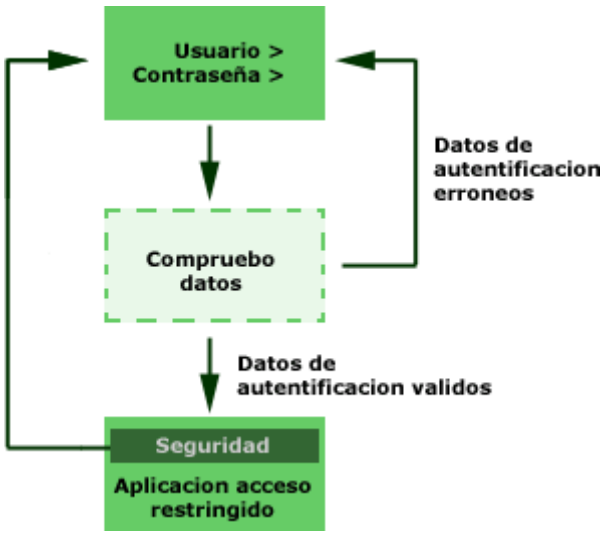
Funcionamiento del sistema de autentificación en PHP

Un sistema de autentificación es un módulo de seguridad para asegurarnos de que el usuario que visita las páginas es quien dice ser. Por supuesto, sabiendo que ese usuario es conocido, podremos darles acceso a más aspectos de la página que si fuese un usuario desconocido.

Esquema de un sistema de autentificación

Vamos a empezar por definir un diagrama para realizar la autentificación de usuario en una página web, que nos servirá para programar luego las páginas ajustándose al diagrama.

En la imagen anterior podemos ver el diagrama, que empieza por la página donde se pide un usuario y contraseña para acceder a la aplicación de acceso restringido.



Los datos de autentificación (usuario y contraseña escritos en la página inicial) se envían a la página dibujada con línea de puntos, que se encarga de hacer una comprobación de dichos datos del usuario. Según los datos de autentificación, se redirecciona al navegador a la página de la aplicación restringida, en caso de que sean correctos, o a la página donde volver a escribir el usuario/contraseña, en caso de que sean incorrectos. Esta página la he dibujado con línea de puntos porque no es una página donde se pare el navegador para nada, sino que sólo es una página de paso que redirecciona a un sitio u otro dependiendo de los datos que reciba.

La aplicación de acceso restringido, aparte de mostrar las funcionalidades que queríamos proteger con usuario contraseña, debe realizar unas comprobaciones de seguridad para saber si se ha pasado con éxito el proceso de autentificación o si se está intentando acceder de manera no permitida a esa página. Esta comprobación la he dibujado como una capa con color verde más oscuro sobre la página de la aplicación. Si no se satisface dicha comprobación (el usuario no se ha autentificado correctamente) se vuelve a la página donde escribir el usuario y la contraseña.

Este es el esquema básico, que espero que se entienda bien. Ahora, veamos algunas

preguntas que podrías hacerte.

¿Por qué hacemos esta comprobación de seguridad dentro de la aplicación?

Podría ser que alguien conociese la URL de la aplicación de acceso restringido y la escribiese directamente sobre la barra de direcciones del explorador, así que hacemos esta comprobación para saber que realmente no se está accediendo sin pasar por la página que comprueba si el usuario/contraseña es correcto

¿Cómo sabemos que ciertamente se ha pasado por la página que comprueba los datos de autentificación?

Esta comprobación la podríamos hacer de varias maneras, así pues, depende de nuestro script de autentificación y el nivel de seguridad que tratemos de implementar. Un ejemplo simple podría ser crear una variable de sesión en la página que comprueba los datos, si es que eran correctos, y en capa se seguridad de las páginas de acceso restringido comprobaríamos si esa sesión está o no definida.

En caso de desear burlar la seguridad, ¿Cómo podría un usuario entrar a la página de la aplicación si no hay enlaces directos y para pasar a ella necesitamos que nos redireccione la página de comprobación del usuario/contraseña?

Pues de diversas maneras, para empezar, el historial de los ordenadores guarda las URL a las que se ha accedido y cualquier persona podría recuperar la URL de nuestra aplicación con acceso restringido. También se podría probar distintas URL que podríamos imaginarnos como posibles para la aplicación y esperar a acertar con el nombre de archivo en algún momento, incluso esta tarea se la podríamos encomendar a un programa para realizar muchas más pruebas. En cualquier caso, nuestra seguridad no se puede quedar en simplemente que los posibles intrusos no conozcan la dirección de la página.

En el manual de referencia de clase ya te he explicado y expuesto el código para realizar un login basándonos en un UNICO usuario y una contraseña, a continuación, te explico cómo hacerlo para en el caso de que el acceso quieras definirlo para MULTIPLES USUARIOS

Autentificación PHP para múltiples usuarios usando MySQL

Vamos a ver las páginas PHP que necesitaríamos para realizar un acceso restringido por clave y contraseña para múltiples usuarios, donde cada uno tenga unos datos de acceso propios.

Nota: Sólo vamos a tratar la página que recoge los datos del usuario (su nombre y contraseña) y comprueba si son correctos, redireccionando a la aplicación segura (si los datos se corresponden con algún usuario de la base de datos), o a la página de entrada (si los datos no correspondían con ningún usuario registrado), evidentemente al basarnos en una tabla, dicha tabla podría tener un campo nuevo con el perfil de acceso, por ejemplo, editor, administrador, autor…, donde según dicho perfil de acceso se le redireccione a la página con las funcionalidades determinadas a dicho perfil

1) **Mediante una tabla en una base de datos**

La base de datos que vamos a utilizar, o bien una simple tabla nueva dentro de tu base de datos ya existente, contendrá una tabla para los usuarios, donde cada uno dispondrá, al menos, de dos campos: un nombre de usuario y una contraseña, los dos de tipo texto.

Tabla usuario	
Nombre del campo	Tipo del campo
nombre_usuario	Texto
clave_usuario	Texto

En una base de datos de usuarios, el nombre de usuario debería ser un valor único, irrepetible para otro usuario, es decir, no podremos tener dos usuarios con el mismo nombre. Por esta razón, el campo nombre_usuario podría ser la clave principal de la tabla, aunque también podríamos haber creado un campo adicional, llamado por ejemplo id_usuario, de tipo autonumérico y colocarlo como clave principal.

Para conseguir no insertar dos usuarios con el mismo nombre de usuario, a la hora de insertarlos en la tabla, comprobaremos que no haya ningún usuario ya introducido con el nombre de usuario que se pretende insertar. Este paso, aunque importante, no lo vamos a ver, pues sólo nos vamos a centrar en decidir si un usuario puede entrar o no en la aplicación, suponiendo que los usuarios se encuentran ya insertados en la base de datos.

Básicamente se trataría de articular, previo al ingreso de usuario, una consulta que pregunte si el usuario introducido del campo de formulario ya existe

En el ejemplo suponemos que utilizamos una base de datos MySQL, sin embargo, cualquier tipo de base de datos podrá servir para unos objetivos como los que nos proponemos.

El funcionamiento del script

El script que se utilizará para decidir si un usuario puede o no entrar en la aplicación es muy sencillo. Simplemente hace una llamada a la base de datos para comprobar si los datos de autentificación escritos por el visitante (usuario y contraseña) corresponden con los de algún usuario. En caso de que así sea, se permite la entrada y de no ser así, se deniega.

Lo primero sería abrir una conexión con la base de datos como venimos haciendo habitualmente y que de modo reducido seria:

```
//conecto con la base de datos
$conn = mysqli_connect("servidor","usuario","password", “base de datos”);
```

Un segundo paso es construir una sentencia SQL que nos permita comprobar si existe o no un usuario con los datos de autentificación introducidos. Utilizamos una simple sentencia SELECT, sobre la tabla de usuarios, donde se extraen usuarios que tengan el mismo nombre de usuario y la contraseña introducidos en la página de acceso.

```
//Sentencia SQL para buscar un usuario con esos datos
     $ssql = "SELECT * FROM usuario WHERE nombre_usuario='$usuario' and
     clave_usuario='$contrasena'";
//Ejecuto la sentencia
```

```
$rs = mysqli_query($conn, $ssql);
```

Si esa sentencia SELECT responde con algún registro encontrado, sabremos que existe un usuario donde sus datos de autentificación corresponden perfectamente con los introducidos. En ese caso podremos realizar las acciones encaminadas a permitir el acceso. Por el contrario, si la sentencia SELECT no encuentra ningún registro, sabremos que no existe un usuario con los datos de autentificación introducidos y por lo tanto, deberemos realizar las acciones encaminadas a restringir el acceso.

```
if (mysqli_num_rows($rs)!=0){
//usuario y contraseña válidos
//defino una sesión y guardo datos session_start(); session_register("autentificado");
      $autentificado = "SI";
      header ("Location: aplicacion.php");
      }else {
            //si no existe le mando otra vez a la portada header("Location:
            index.php?errorusuario=si");
}
```

Las acciones para restringir o permitir el acceso son exactamente iguales a las que veníamos utilizando en el script de control sin utilizar la base de datos. Así que no vamos a comentarlas más, están en el manual aportado.

El código completo del ejemplo sería el siguiente.

```
<?
//conecto con la base de datos
$conn = mysqli_connect("servidor","usuario","password", “basedeDatos”);
//Sentencia SQL para buscar un usuario con esos datos
$ssql = "SELECT * FROM usuario WHERE nombre_usuario='$usuario' and
clave_usuario='$contrasena'";

//Ejecuto la sentencia
      $rs = mysql_query($ssql,$conn);

//vemos si el usuario y contraseña es válido
//si la ejecución de la sentencia SQL nos da algún resultado
//es que sí que existe esa combinación usuario/contraseña
      if (mysql_num_rows($rs)!=0){
      //usuario y contraseña válidos
      //defino una sesion y guardo datos
      session_start();
      session_register("autentificado");
      $autentificado = "SI";
      header ("Location: aplicacion.php");
```

```
      }else {
//si no existe le mando otra vez a la portada
      header("Location: index.php?errorusuario=si");
}
      mysqli_free_result($rs); mysqli_close($conn);
?>
```

Nota: Es importante destacar que esta página no debería contener ningún tipo de texto antes de la apertura de código PHP, ni tan siquiera saltos de línea. Esto es debido a que al final se realiza una redirección y este tipo de instrucciones solamente se puede ejecutar si no se ha escrito todavía ningún carácter en el cuerpo. Para ser más específicos, este es el error que obtenemos si escribimos antes en la página de enviar las cabeceras:
Warning: Cannot add header information - headers already sent by (output started at /htdocs/ejemplos/autentif-php_bbdd/control.php:2) in /htdocs/ejemplos/autentif-php_bbdd/control.php on line 26

2) **Otra posibilidad de codificación**

Autenticar usuario y guardar en una cookie con PHP

¡OJO!, código revisado, pero sin verificar.

Vamos a crear un sistema para autenticar usuarios con PHP, con la particularidad que este sistema va a ofrecer al visitante la opción de guardar su usuario, para que la página lo recuerde en sucesivos accesos y no tenga que volver a autenticarse. El usuario se guardará en una cookie para que el navegador pueda recordarlo en sus distintas visitas.

Esta es una opción muy útil para que el visitante no tenga que estar todo el tiempo autenticándose, con su usuario y contraseña, cada vez que accede a la página web. Seguro que es una opción que habremos visto en un montón de sitios web.

En este ejemplo no vamos a realizar una autenticación muy elaborada, sino una muy simple, para facilitar el desarrollo. Luego la complejidad la aportará la parte de guardar el usuario en una cookie, que no es difícil de hacer, pero sí requiere de nuevos conocimientos que aplicar.

Explicación de almacenar el usuario en una cookie

Primero vamos a explicar con palabras el modelo de trabajo que vamos a aplicar para almacenar el usuario en una cookie. No he investigado cómo lo podrán hacer esto otras personas en otros desarrollos, pero creo que he ingeniado una forma adecuada para hacerlo.

Digo esto porque lo primero que se me ocurrió fue meter el nombre de usuario y la clave en unas cookies. Sería simplemente crear un par de cookies en el sistema del usuario con esas dos variables. Pero luego pensando, no me parecía muy atractiva la posibilidad de incluir esa información sensible en unas cookies en el ordenador del usuario, por su hubiera alguna persona que pudiera leerlas, copiarlas y utilizarlas en otro ordenador. Igual no me debería preocupar por ello, pero en cualquier caso no me gustaba la posibilidad de almacenar el nombre de usuario y contraseña, sino almacenar otro tipo de información menos crítica.

Entonces lo que se me ocurrió es almacenar el identificador del usuario. Pero claro, si alguien conseguía crear una cookie en el sistema con cualquier identificador de usuario, podría acceder a la cuenta de ese usuario. Así que había que aplicar otra estrategia adicional para asegurar que ese identificador de usuario no se pueda reproducir. Finalmente, decidí generar un número aleatorio cuando el usuario se conecta a la página y se autentica correctamente, y almacenarlo en dos sitios, primero en el registro del usuario en la base de datos y luego en la cookie.

Así, cuando el visitante decide que quiere que el sitio web le recuerde su cuenta de usuario, para no tener que volver a autenticarse en siguientes accesos, se guardan dos cosas en las cookies del navegador: su identificador de usuario y una marca aleatoria (dicha marca también se almacena en la base de datos, asociada a su usuario). En siguientes accesos primero se comprueba si existen esas cookies en el navegador del visitante y si el conjunto de identificador de usuario y la marca aleatoria almacenados en la cookie coincide con lo que tenemos en la base de datos.

Dicho de otra manera, en la cookie guardamos el identificador de usuario. Además, generamos un número aleatorio que guardamos en dos sitios: 1) en la tabla de usuario, en el registro correspondiente al usuario que desea que se le recuerde la clave y 2) en una cookie. Luego cuando el usuario se conecta de nuevo, no sólo se comprueba que tenga la cookie con el identificador de usuario, sino que tenga la otra cookie con la marca aleatoria y que sea la misma que tenemos en la base de datos para ese mismo usuario.

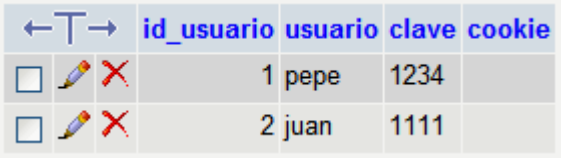
Espero que el sistema se pueda entender. No obstante, espero clarificarlo aún más a medida que explique el código PHP y la base de datos que vamos a utilizar.

Tabla de usuario

En la tabla de usuario que hemos creado para este ejemplo tenemos 4 campos:

- Identificador de usuario
- Nombre de usuario
- Clave de acceso
- Marca aleatoria que se ha metido en la cookie

Para el ejemplo hemos creado la tabla e insertado un par de usuarios para hacer pruebas. Tendrá una forma como esta:



←T→	id_usuario	usuario	clave	cookie
☐ ✎ ✕	1	pepe	1234	
☐ ✎ ✕	2	juan	1111	

Código de la página PHP

Ahora voy a explicar por partes el código PHP para autenticar al usuario y guardar la información de acceso en la cookie, así como la parte de comprobar si el usuario tenía la cookie con su acceso guardado en el ordenador.

Primero vamos a empezar mostrando el formulario HTML:

```
<form action="prueba-cookies.php" method="post">
Usuario: <input type="text" name="usuario">
<br>
Clave: <input type="text" name="clave">
<br>
<input type="checkbox" name="guardar_clave" value="1"> Memorizar el usuario en este ordenador
<br>
<input type="submit" value="Entrar">
</form>
```

Como vemos, tiene los campos para escribir el nombre de usuario y la clave y un campo checkbox adicional para que el usuario marque si quiere que su acceso se guarde en su ordenador.

Ahora voy a mostrar el código que utilizaríamos para recibir por post, del formulario de autentificación, el nombre de usuario y contraseña. Este código también tiene que detectar si el usuario quería que se guardase su acceso en el ordenador.

```
//debería comprobar si el usuario es correcto
$ssql = "select * from usuario where usuario = '" . $_POST["usuario"] . "' and clave='" . $_POST["clave"] . "'";
//echo $ssql;
$rs = mysqli_query($ssql);
if (mysqli_num_rows($rs)==1){
//TODO CORRECTO!! He detectado un usuario
$usuario_encontrado = mysqli_fetch_object($rs);
//ahora debo ver si el usuario quería memorizar su cuenta en este ordenador if ($_POST["guardar_clave"]=="1"){
//es que pidió memorizar el usuario
//1) creo una marca aleatoria en el registro de este usuario
//alimentamos el generador de aleatorios mt_srand (time());
//generamos un número aleatorio
$numero_aleatorio = mt_rand(1000000,999999999);
//2) meto la marca aleatoria en la tabla de usuario
$ssql = "update usuario set cookie='$numero_aleatorio' where id_usuario=" . $usuario_encontrado->id_usuario; mysql_query($ssql);
//3) ahora meto una cookie en el ordenador del usuario con el identificador del usuario y la cookie aleatoria setcookie("id_usuario_dw", $usuario_encontrado->id_usuario , time()+(60*60*24*365)); setcookie("marca_aleatoria_usuario_dw", $numero_aleatorio, time()+(60*60*24*365));
}
echo "Autenticado correctamente";
//header ("Location: contenidos_protegidos_cookie.php");
}else{
echo "Fallo de autenticación!";
```

```
echo "<p><a href='prueba-cookies.php'>Volver</a>";}
```

Para comprobar si los datos de autenticación que recibimos por el formulario son correctos, hacemos una sentencia SQL. La ejecutamos y si nos da como resultado que tenemos un registro encontrado en la tabla de usuarios, es que el nombre de usuario y clave corresponden con el de algún usuario.

Luego con la línea

```
if ($_POST["guardar_clave"]=="1"){
```

Comprobamos si el visitante había pedido que se almacenase la clave en su ordenador. Entonces hay que generar las cookies correspondientes, que habíamos comentado anteriormente en este artículo.

Lo hacemos en tres pasos:

- Genero un número aleatorio para que nos sirva de marca.
- Inserto la marca aleatoria en la tabla de usuarios, haciendo un update en el registro del usuario autenticado que habíamos detectado anteriormente.
- Genero y coloco en el ordenador del usuario las dos cookies para guardar su acceso en el navegador: el identificador del usuario y la marca aleatoria. Hemos creado las cookies para que se almacenen durante un año en el ordenador del usuario.

Nota: Es importante señalar que, para colocar o crear cookies en el navegador del visitante, debemos hacerlo antes de que se hayan enviado las cabeceras de http, es decir, antes de haber escrito cualquier texto en la página. Si no, nos podría dar un error de http headers already sent.

Por último, veamos el código PHP para ver si detectamos las cookies en el navegador de un usuario autenticado anteriormente en el sistema y guardado en el ordenador del usuario.

```
//primero tengo que ver si el usuario está memorizado en una cookie
if (isset($_COOKIE["id_usuario_dw"]) && isset($_COOKIE["marca_aleatoria_usuario_dw"])){
//Tengo cookies memorizadas
//además voy a comprobar que esas variables no estén vacías
if ($_COOKIE["id_usuario_dw"]!="" || $_COOKIE["marca_aleatoria_usuario_dw"]!=""){
//Voy a ver si corresponden con algún usuario
$ssql = "select * from usuario where id_usuario=" . $_COOKIE["id_usuario_dw"] . " and
cookie='" .
$_COOKIE["marca_aleatoria_usuario_dw"] . "' and cookie<>''";
$rs = mysql_query($ssql);
if (mysql_num_rows($rs)==1){
echo "<b>Tengo un usuario correcto en una cookie</b>";
$usuario_encontrado = mysql_fetch_object($rs);
echo "<br>Eres el usuario número " . $usuario_encontrado->id_usuario . ", de nombre " .
$usuario_encontrado->usuario;
//header ("Location: contenidos_protegidos_cookie.php");
}
}
}
```

Como primer paso compruebo si existen las cookies con el identificador del usuario y la mencionada marca aleatoria. Además, hacemos una comprobación adicional para ver si alguna de las dos cookies contiene un string vacío, porque en ese caso no están guardadas correctamente y no nos sirven.

Si todo ha ido bien, miramos en la base de datos si el usuario con identificador determinado en la cookie tiene la marca aleatoria igual que la que tenía la cookie del navegador del visitante.

Además, en la consulta en la base de datos también nos aseguramos que la marca aleatoria sea distinta de "", porque entonces es un usuario que nunca había pedido que se guardasen sus datos en el ordenador.

Si esa consulta daba un registro, es que corresponde con un usuario que se había almacenado en el ordenador y es el usuario que estaba autenticado anteriormente y guardado su acceso.

Conclusión

Hasta aquí he comentado todo lo que necesitamos saber para crear la infraestructura para que la página web recuerde la clave del usuario y no tenga que autenticarse cada vez que accede al sitio. El código que hemos mostrado podría completarse con una serie de mejoras o personalizaciones para adaptarlo a nuestras necesidades, pero seguro que sirve de guía para el interesado.

Ahora presento el código completo de la página de este ejemplo:

```
<?
//conecto con la base de datos
$conn = mysqli_connect("servidor","usuario","clave,"base de datos");

//primero tengo que ver si el usuario está memorizado en una cookie
if (isset($_COOKIE["id_usuario_dw"]) && isset($_COOKIE["marca_aleatoria_usuario_dw"])){
//Tengo cookies memorizadas
//además voy a comprobar que esas variables no estén vacías
if ($_COOKIE["id_usuario_dw"]!="" || $_COOKIE["marca_aleatoria_usuario_dw"]!=""){
//Voy a ver si corresponden con algún usuario
$ssql = "select * from usuario where id_usuario=" . $_COOKIE["id_usuario_dw"] . " and
cookie='" .
$_COOKIE["marca_aleatoria_usuario_dw"] . "' and cookie<>''";
$rs = mysqli_query($conn,$ssql);
if (mysqli_num_rows($rs)==1){
echo "<b>Tengo un usuario correcto en una cookie</b>";
$usuario_encontrado = mysqli_fetch_object($rs);
echo "<br>Eres el usuario número " . $usuario_encontrado->id_usuario . ", de nombre " .
$usuario_encontrado->usuario;
//header ("Location: contenidos_protegidos_cookie.php");}}}
```

```
if ($_POST){
//es que estamos recibiendo datos por el formulario de autenticación (recibo de $_POST)
//debería comprobar si el usuario es correcto
$ssql = "select * from usuario where usuario = '" . $_POST["usuario"] . "' and clave='"
. $_POST["clave"] . "'";
//echo $ssql;
$rs = mysql_query($ssql);
if (mysql_num_rows($rs)==1){
//TODO CORRECTO!! He detectado un usuario
$usuario_encontrado = mysql_fetch_object($rs);
//ahora debo de ver si el usuario quería memorizar su cuenta en este ordenador if
($_POST["guardar_clave"]=="1"){
//es que pidió memorizar el usuario
//1) creo una marca aleatoria en el registro de este usuario
//alimentamos el generador de aleatorios
mt_srand (time());
//generamos un número aleatorio
$numero_aleatorio = mt_rand(1000000,999999999);


//2) meto la marca aleatoria en la tabla de usuario


$ssql = "update usuario set cookie='$numero_aleatorio' where id_usuario=" .
$usuario_encontrado->id_usuario;
mysqli_query($conn,$ssql);
//3) ahora meto una cookie en el ordenador del usuario con el identificador del usuario
y la cookie aleatoria
setcookie("id_usuario_dw", $usuario_encontrado->id_usuario , time()+(60*60*24*365));
setcookie("marca_aleatoria_usuario_dw", $numero_aleatorio, time()+(60*60*24*365));
}
echo "Autenticado correctamente";
//header ("Location: contenidos_protegidos_cookie.php");


}else{
echo "Fallo de autenticación!";
echo "<p><a href='prueba-cookies.php'>Volver</a>";
}


}else{
?>
```

Ahora la parte de HTML

```
<!DOCTYPE HTML >
<html>
<head>
```

```
<title>cookies para detectar usuario</title>
</head>
<body>
<form action="prueba-cookies.php" method="post"> Usuario: <input type="text"
name="usuario">
<br>
Clave: <input type="text" name="clave">
<br>
<input type="checkbox" name="guardar_clave" value="1"> Memorizar el usuario en este
ordenador
<br>
<input type="submit" value="Entrar">
</form>
<br>
<br>
<b>Usuarios válidos:</b>
<br>
<br>
User: pepe
<br> Clave: 1234
<br>
<br> User: juan
<br> Clave: 1111
</body>
</html>
<?
}
?>
```