



UTN.BA FACULTAD
REGIONAL
BUENOS AIRES
SECRETARÍA DE EXTENSIÓN UNIVERSITARIA FRBA UTN

**Centro de
e-Learning**

EXPERTO UNIVERISTARIO EN MySQL Y PHP NIVEL INTERMEDIO



www.sceu.frba.utn.edu.ar/e-learning



Módulo 2

Mas coceptos sobre Mysql y PHP



Variables de session y cookies.



Presentación de la Unidad:

En esta unidad veremos dos tipos de variables de PHP, las variables de sesión y las cookies.

Vamos a ver cual es la utilidad de las mismas a través de ejemplos prácticos.



Objetivos:

- ❖ Utilidad de las cookies.
- ❖ Utilidad de las variables de session.



Temario:

Cookies

Variables de session



CONSIGNAS PARA EL APRENDIZAJE COLABORATIVO

En esta Unidad los participantes se encontrarán con diferentes tipos de consignas que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:



1. Los foros asociados a cada una de las unidades.
2. La Web 2.0.
3. Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen las actividades sugeridas y compartan en los foros los resultados obtenidos.

COOKIES

Las cookies son un mecanismo que utilizan los servidores web para guardar información en el ordenador del usuario y recuperarla cada vez que el navegador les pide una página. La información se guarda en el ordenador del usuario en forma de texto y está formada por parejas (nombre de la cookie y valor de la cookie).

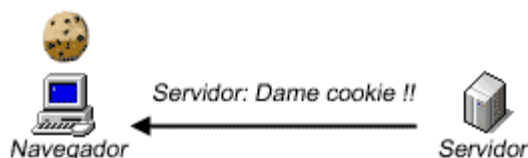
Esta información se utiliza para numerosos fines (autenticación, selección de preferencias, items seleccionados en un carrito de la compra, etc.), siempre con la intención de identificar al usuario y personalizar las páginas.

Funcionamiento

La cookie es enviada al navegador desde el servidor y si este la acepta permanece en él.



Las páginas piden la cookie al navegador...



El navegador las envía, permitiendo la identificación del usuario por parte del servidor.



PHP soporta transparentemente cookies HTTP.

Las Cookies, como dijimos, son un mecanismo que sirve para almacenar datos en el navegador del usuario remoto, para así poder identificar al usuario cuando vuelva.

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. Esta limitación es la misma a la de la función `header()`. Se pueden usar las funciones de almacenamiento intermedio

del resultado para retrasar el resultado del script hasta que hayas decidido mandar o no una cookie o cabecera.

Cualquier cookie enviada a ti desde el cliente, automáticamente se convertirá en una variable PHP igual que ocurre con los métodos de datos GET y POST, dependiendo de las variables de configuración `register_globals` y `variables_order`.

En PHP 4.1.0 y posteriores, la matriz global `$_COOKIE` será siempre actualizada con cualquier cookie mandada por el cliente.

`$HTTP_COOKIE_VARS` es también actualizada en versiones anteriores de PHP cuando la variable de configuración `track_vars` esté activada. (Siempre activada a partir de PHP 4.0.3.)

Crear cookies

Las cookies se crean cuando el servidor se lo pide al navegador. Cuando el servidor envía una página al navegador, puede incluir en las cabeceras de la respuesta HTTP la petición de creación de una o varias cookies. El navegador crea la cookie, guardando no sólo el nombre y el valor de la cookie, sino el nombre del servidor (del dominio) que ha creado la cookie.

En PHP, las cookies se crean mediante la función `setcookie()`.

Ejemplo:

```
<?php  
setcookie("nombre", "Veronica");  
?>
```

Si se quiere guardar en una cookie una matriz de datos, es necesario crear cada elemento de la matriz en una cookie distinta.

Ejemplo:

```
<?php  
setcookie("datos[nombre]", "Veronica");  
setcookie("datos[apellido]", "Piñeyro");  
?>
```

Es muy importante que los nombres de las cookies no coincidan con los nombres de controles de los formularios, porque PHP incluye los valores de las cookies en la matriz \$_REQUEST. Es decir, que si el nombre de una cookie coincide con el nombre de un control, en \$_REQUEST sólo se guardará el valor de la cookie, no el del control.

Hay que tener la precaución de utilizar la función setcookie() antes de empezar a escribir el contenido de la página, porque de lo contrario, PHP producirá un aviso y no se creará la cookie. El motivo es que las cookies se crean mediante cabeceras de respuesta HTTP y las cabeceras se envían antes del texto de la página.

Es decir, cuando PHP encuentra una instrucción que escribe texto, cierra automáticamente la cabecera; si a continuación PHP encuentra en el programa la función setcookie(), da un aviso porque ya se han enviado las cabeceras y no se crea la cookie.

Sintaxis Completa

```
setcookie (string Nombre [, string Valor [, int Expire [, string Path [, string Dominio [, int Secure]]]])
```

Todos los argumentos excepto el nombre son opcionales.

- **Nombre.** Nombre de la cookie. Si creamos una cookie solamente con el nombre, en el cliente se eliminara la cookie que exista con ese nombre. También podemos reemplazar cualquier argumento con una cadena vacía ("").

- **Value.** Valor que almacenará la cookie en el cliente.
- **Expire.** El argumento expire es un argumento entero que indica la hora en que se eliminara la cookie en el formato de hora que devuelven las funciones UNIX time() y mktime(). Normalmente se usa time() + N. segundos de duración, para especificar la duración de una cookie.
- **Path.** Subdirectorío en donde tiene valor la cookie.
- **Dominio.** Dominio en donde tiene valor la cookie. Si ponemos como dominio www.domain.com la cookie no se transmite para domain.com, mientras que si ponemos domain.com la cookie se transmite tanto para domain.com como para www.domain.com
- **Secure.** El argumento secure indica que la cookie solo se transmitirá a través de una conexión segura HTTPS.

Ejemplo:

```
setcookie("usuario", Juan", time()+7200,"/","mipagina.com");
```

En este ejemplo establecemos una cookie de nombre usuario que contiene el valor Juan, que dura 2 horas (7200 segundos) válida para todo el dominio mipagina.com

Ejemplo:

En este ejemplo vamos a ver cómo establecer una cookie y cómo se recupera el valor establecido. Para ello pediremos al usuario que introduzca su nombre, que guardaremos en una cookie.

Primero pedimos al usuario que introduzca el valor de su nombre, usamos un formulario que procesará la página pagina1.html

```
<html>
<head>
<title> Página 1</title>
</head>
<body>
<h1>Ejemplo de uso de cookie</h1> Introduzca su nombre:
<form action=" pagina2.php" method="GET">
<input type="text" name="nombre"/><br/>
```

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.

UTN - FRBA. Secretaría de Cultura y Extensión Universitaria

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

```
<input type="submit" value="enviar"/>
</form>
</body>
</html>
```

Se establece la cookie 'usuario' con el valor introducido anteriormente, y cuya duración es una hora.

```
<?php
setcookie("usuario", $_GET['nombre'], time()+3600,"/","");
?>
<html>
<head>
<title>Página 2</title>
</head>
<body>
<h1>Ejemplo de uso de cookie</h1>
Se ha establecido una cookie de nombre <strong>usuario</strong> con el valor: <
strong><?php print
$_GET['nombre']; ?></strong> que será válida durante 1 hora.
<a href="pagina3.php">Ir a pagina3.php</a>
</body>
</html>
```

En este ejemplo vemos lo fácil que es recuperar el valor de la cookie establecida anteriormente.

```
<html>
<head>
<title> Página 3</title>
</head>
<body>
<h1>Ejemplo de uso de cookie</h1>
Se ha establecido la cookie de nombre <strong>usuario</strong> vale: <strong>
<?php print
$_COOKIE['usuario']; ?></strong>
</body>
</html>
```

Críticas a las cookies

Aunque las cookies no son programas y por tanto no pueden contener virus ni dañar

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.

UTN - FRBA. Secretaría de Cultura y Extensión Universitaria

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

nuestras computadoras, mucha gente desconfía de las cookies, hasta el punto que todos los navegadores incluyen opciones para impedir totalmente la creación de cookies o permitirla únicamente a sitios autorizados. Lógicamente, impedir la creación de cookies puede complicar la visita de algunos sitios que basan su funcionamiento en ellas.

Algunos motivos por los que mucha gente desconfía de las cookies son los siguientes:

- Muchos sitios guardan cookies de forma indiscriminada, por motivos que no tienen nada que ver con facilitar al usuario la visita del sitio, sino para hacer un seguimiento de sus hábitos de uso de Internet, seguimiento que se suele hacer sin el consentimiento del usuario.
- Además, al estar las cookies vinculadas al servidor que las crea, el tener en el ordenador una cookie de un dominio puede dar a entender que se ha visitado ese dominio y dar pie a malentendidos.

Ejemplo: Contador

```
<?php
$numero=$_COOKIE["visitante"];
$numero+=1; setcookie("visitante",$numero,time()+86400);
if($numero==1){print "Es la primera vez que visitas esta página";}
if($numero>1){print "Es la $numero º vez que visitas esta página";}
?>
```

SESIONES

¿Qué son las sesiones?

Básicamente una sesión es la secuencia de páginas que un usuario visita en un sitio web, desde que entra en nuestro sitio, hasta que lo abandona.

El término sesión en PHP, session en inglés, se aplica a esta secuencia de navegación. Para ello crearemos un identificador único que asignamos a cada una de estas sesiones de navegación. A este identificador de sesión se le denomina, comúnmente, como la sesión.

El proceso en cualquier lenguaje de programación podría ser algo así:

¿Existe una sesión?
Si existe, la retomamos
Si no existe, creamos una nueva
Genera un identificador único

Y para que no perdamos el hilo de la navegación del usuario deberemos asociar esta sesión a todas las URLs y acciones de formulario. Podemos también crear un cookie que incluya el identificador de sesión, pero es conveniente recordar que la disponibilidad o no de las cookies depende del usuario, y no es conveniente confiar solo en lo que un usuario pueda o no tener habilitado.

Desde la versión PHP4, toda la gestión de sesiones la hace el mismo PHP y no depende de librerías externas, como ocurría en versiones anteriores.

Las sesiones son un método seguro y eficaz de guardar y mantener datos del usuario durante toda su visita. Podemos guardar por ejemplo una variable que diga si está identificado en nuestro sistema o no y si lo está, también podemos guardar sus datos.

Como funcionan

Para iniciar una sesión lo haremos con la función `session_start()`; y luego, las variables que queramos mantener durante toda la visita del usuario las guardaremos en la variable `$_SESSION`.

Siempre que queramos utilizar las sesiones deberemos llamar antes a `session_start()`; y siempre antes que cualquier salida. Es decir, antes que "<html>" e incluso antes que cualquier echo o función que imprima en pantalla.

Ejemplo sobre el funcionamiento de las sesiones.

Crearemos tres páginas, "uno.php", "dos.php" y "tres.php". En la primera pondremos lo siguiente:

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.

UTN - FRBA. Secretaría de Cultura y Extensión Universitaria

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar



```
<?php session_start();  
$_SESSION['usuario'] = 'Veronica';  
?>  
<html>  
<head>  
<title>Prueba de sesión</title>  
</head>  
<body>  
<a href='dos.php'>Ir a la segunda página</a>  
</body>  
</html>
```

En esta primera página iniciamos la sesión y guardamos en la sesión el valor 'Veronica' con nombre 'usuario'.

En la página "dos.php" pondremos el siguiente código:

```
<?php session_start();  
echo $_SESSION['usuario'];  
?>
```

Si abrimos uno.php en el navegador y hacemos click en el enlace, iremos a la página dos.php donde veremos 'Veronica' si todo ha ido bien.

En la página "tres.php" pondremos el mismo código que en "dos.php" y efectivamente veremos también el nombre de usuario.

Esto es la base de las sesiones, y lo podemos aplicar por ejemplo para identificar a un usuario y mantenerlo identificado durante toda su visita.

Trabajando con sesiones siempre tendremos que tener en cuenta:

- Poner siempre al principio session_start();
- Nunca poner nada que imprima algo en pantalla antes que session_start()
- Usar siempre la variable \$_SESSION para manejar las variables de sesión.

Funciones de Sesiones

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.

UTN - FRBA. Secretaría de Cultura y Extensión Universitaria

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

`session_cache_expire` — Devuelve la caducidad de la caché actual
`session_cache_limiter` — Obtener y/o establecer el limitador de caché actual
`session_commit` — Alias de `session_write_close`
`session_decode` — Decodifica la información de sesión desde una cadena
`session_destroy` — Destruye toda la información registrada de una sesión
`session_encode` — Codifica la información de la sesión actual como una cadena
`session_get_cookie_params` — Obtener los parámetros de la cookie de sesión
`session_id` — Obtener y/o establecer el id de sesión actual
`session_is_registered` — Averiguar si una variable global está registrada en una sesión
`session_module_name` — Obtiene y/o establece el módulo de sesión actual
`session_name` — Obtener y/o establecer el nombre de la sesión actual
`session_regenerate_id` — Actualiza el id de sesión actual con uno generado más reciente
`session_register` — Registrar una o más variables globales con la sesión actual
`session_save_path` — Obtener /establecer la ruta de almacenamiento de la sesión actual
`session_set_cookie_params` — Establecer los parámetros de la cookie de sesión
`session_set_save_handler` — Establece funciones de almacenamiento de sesiones a nivel de usuario
`session_start` — Inicializar información de sesión
`session_unregister` — Deja de registrar una variable global de la sesión actual
`session_unset` — Libera todas las variables de sesión
`session_write_close` — Escribir información de sesión y finalizar la sesión

Como ya nos ha pasado con PHP, esta lista contiene algunas funciones que han sido declaradas obsoletas en la última versión de PHP (como por ejemplo `session_register`), como ya vimos también, esto no significa que no sigan funcionando.

Les recomiendo siempre estar atentos a:
<http://www.php.net/manual/es>

Profundicemos algunas de ellas:

session_start()

Crea una sesión -o continua con la actual- en función del identificador de sesión pasada por una variable GET, POST o por una cookie.
Devuelve siempre TRUE

session_name()

Recoge el nombre de la sesión. Si no se asigna uno de forma explícita, utiliza como nombre de sesión el contenido de la directiva `session.name` del fichero de configuración `php.ini`. (Por defecto ese nombre suele ser `PHPSESSID`)

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.

UTN - FRBA. Secretaría de Cultura y Extensión Universitaria

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

session_name('nombre')

Esta función te permite asignar un nuevo nombre a la sesión actual.

Hay que tener en cuenta que si se cambia de página manteniendo el mismo identificador de sesión (sin cambiar de sesión) esta función (con el nombre correspondiente) debe ser escrita en cada página, debe estar siempre antes que la llamada a la función session_start().

session_cache_limiter()

El limitador de caché controla las cabeceras HTTP de control del caché enviadas al cliente. Estas cabeceras determinan las reglas por las que el contenido de la página puede ser guardado en el caché local del cliente.

Esta opción viene configurada por defecto en las directivas de configuración del fichero php.ini y por defecto suele aparecer activada la opción nocache.

session_cache_limiter ('nocache,private').

Evita el almacenamiento en el caché del cliente.

Al igual que la función session_name, si se utiliza esta función deberás colocarla antes de session_name e igual que en aquel caso deberás repetirla también en cada página.

session_id()

ID de una Sesión. Se usa para proporcionar (leer) el número ID de sesión que se ha inicializado. También se usa esta función para cambiar el ID de la sesión actual.

```
echo ("TU ID: ". session_id() );
```

Salida:

TU ID: 85a81f7ae2149ab05b8a4e9a307c2ce7

\$_SESSION

Es un arreglo (matriz) asociativa que contiene las variables de sesión disponibles en la aplicación web actual.

```
$_SESSION["autorizado"] = "OK"
```

Propagación de las sesión

La verdadera utilidad de las sesiones es su posibilidad de ser propagadas, es decir que tanto el identificador de sesión como los valores de las variables de sesión puedan irse pasando (propagando) de una página a otra sin necesidad de recurrir al uso de formularios.

Esto sería como entrar a un bar, que nos den un ticket con su número impreso (el identificador de sesión) donde se anotará todo lo consumido, con él en la mano podemos pedir cualquier cosa dentro, podemos tomarnos una cerveza (la anotarán en el ticket) y luego irnos a utilizar las instalaciones del bar (cosa normal habiendo tomado cerveza, esto obviamente sin salir del lugar), luego volver a la barra y pedir otra o algo de comer, o ambas cosas, etc. todo esto, será agregado al ticket.

En cada sección -equivalente a distintas páginas web del mismo sitio- te harán las anotaciones correspondientes siempre en el mismo ticket, es decir, en cada uno de esos movimientos irás propagando la sesión por las diferentes secciones.

En PHP ocurre lo mismo que en esos bares... se te permite propagar las sesiones pero tratando de controlar tus movimientos y tratando de evitar que algún desaprensivo entre sin ticket o se vaya sin pagar.

Lo normal y lo más fácil es que esa propagación se haga mediante cookies pero como es el usuario el que tiene la posibilidad de desactivar la opción no aceptar cookies sin que nosotros podamos hacer nada al respecto, los desarrolladores de PHP han previsto una opción alternativa para esa situación que es la propagación a través de la URL.

Propagación con cookies activados

Si queremos que las sesiones se propaguen únicamente cuando está activada la opción aceptar cookies en el navegador del cliente lo primero que hay que comprobar es un pequeño detalle de la configuración de tu PHP.

Vamos a `info.php` y nos fijamos si `session.use_trans_sid` tiene valor 1 ó 0.

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.

UTN - FRBA. Secretaría de Cultura y Extensión Universitaria

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

Si ese valor es 1 indicaría que PHP ha sido compilado con la opción --enable-trans-sid, o lo que es lo mismo que está activada la propagación en modo transparente. Si eso es así para que la sesión se propague bastará con que hagas la llamada a la nueva página siguiendo el método habitual, es decir:

```
<a href="pagina.php"> y que esa nueva página tenga al principio del todo el famoso script
<?php session_start();
.....
?>
```

Es importante saber que session_cache_limiter no es imprescindible (solo hay que ponerlo si se quiere establecer un comportamiento de la caché distinto al configurado en php.ini) pero hay que tener en cuenta que ese parámetro no se propaga y que si no lo

especificamos al comienzo de cada página no se mantendrá el de la página anterior y utilizará el valor que tenga configurado en el archivo php.ini.

Algo parecido ocurre con session_name. Si al iniciar la sesión le ponemos un nombre a la sesión para que la sesión se propague debemos poner en el famoso script anterior la función session_name y además debe contener el mismo nombre que le habíamos adjudicado a la sesión en la página de procedencia.

Si no le ponemos nombre a la sesión, es decir dejamos que lleve el nombre por defecto (normalmente PHPSESSID) no hará falta que utilicemos en ningún sitio session_name.

¿Y si no está activado el modo transparente o se tiene los cookies desactivados?

En este caso tendremos que trabajar un poco más. Para no tener que programar manualmente el PHPSESSID en nuestros script y no preocuparnos por que el cliente no acepte las cookies de nuestra session deberemos tener en nuestro php.ini las siguientes directivas activadas (1) o desactivadas (0)

```
session.use_cookies=0 session.use_trans_sid=1
```

Si no es así o no tenemos acceso al php.ini deberemos poner en todos nuestros script que usen sesiones lo siguiente:

Código PHP:

```
ini_set("session.use_cookies","0"); //desactivamos el uso de cookies para nuestras sessions
```

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.

UTN - FRBA. Secretaría de Cultura y Extensión Universitaria

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

`ini_set("session.use_trans_sid","1");` //activamos el uso de la url para enviar el SID de nuestra sesión

`ini_set`

`ini_set (string $varname, string $newvalue)`

Establece el valor de la directiva de configuración dada. El nuevo valor establecido se mantendrá durante la ejecución del script y se restaurará cuando acabe el mismo.

varname

Nombre de la directiva a modificar. No todas las directivas pueden ser modificadas con `ini_set()`. Hay una lista de todas las directivas disponibles en:

<http://www.php.net/manual/es/ini.list.php>

newvalue

Nuevo valor para la directiva.

Valores devueltos

Devuelve el valor anterior en caso de éxito, FALSE en caso de error.

Ejemplo sencillo y práctico de sesiones:

En este simple ejemplo que tenemos dividido en dos archivos, vemos que `acceso.php` es el archivo que nos creará una sesión llamada `access`, con un valor true o verdadero para luego mostrar OK. Luego abrimos `verificar.php` y nos mostrará que tenemos el acceso permitido, ya que hemos abierto `acceso.php` primero.

Si no hemos abierto `acceso.php`, nos dará error `verificar.php`, ya que no tenemos la sesión creada.

Esto se puede implementar para sistemas de usuarios en el que no se quiera utilizar COOKIES , se puede reemplazar por Sesiones para más seguridad.

acceso.php

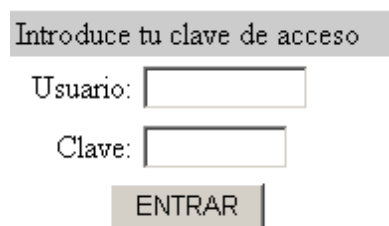
```
<?php session_start();  
$_SESSION[access] = true; echo "OK";  
?>
```

Archivo: verificar.php

```
<?php session_start();  
if($_SESSION[access]==true) {  
    echo "OK, tienes el acceso permitido";  
}  
else {  
    echo "Error, no tienes permiso."; sesión_destroy();  
}  
?>
```

Un ejemplo un poco más completo, en este caso, no solo chequea, sino que de estar OK me lleva a una determinada página y en caso contrario a otra.

index.php



```
<html>  
<head>  
<title>index.html</title>  
</head>  
<body>  
<form action="control.php" method="post">  
<table>  
<tr>  
<td colspan="2" align="center">  
<?php if ($_GET["errorusuario"]=="si"){?>  
<p><b>datos incorrectos</b></p>  
<?php }else{?>  
<p>Introduce tu clave de acceso</p>  
<?php }?></td>  
</tr><br/>
```

```
<tr>
<td align="right">Usuario:</td>
<td><input type="text" name="usuario" size="8" maxlength="50"/></td>
</tr><br/>
<tr>
<td align="right">Clave:</td>
<td><input type="password" name="contrasena" size="8" maxlength="50"/></td>
</tr><br/>
<tr> <td> <input type="submit" value="entrar"/></td>
</tr>
</table>
</form>
</body>
</html>
```

control.php

```
<?php session_start();
//verificamos si el usuario y contraseña es válido
if ($_POST["usuario"]=="coquito" && $_POST["contrasena"]=="123"){
//usuario y contraseña válidos
//crear variable para la sesión
$_SESSION["autenticado"]="1";
$_SESSION["user"]=$_POST["usuario"];
$_SESSION["pass"]=$_POST["contrasena"]; header ("location: aplicacion.php");
}else {
//si no existe, ir a la página de inicio
header("location: index.php?errorusuario=si");
}
?>
```

seguridad.php

```
<?php
//inicio la sesión
session_start();
//comprueba que el usuario esta autenticado
if ($_SESSION["autenticado"] != "1") {
//si no existe, se dirige a la página de inicio
header("location: index.php");
//salimos del script
exit();
}
?>
```



aplicacion.php

```
<?php include ("seguridad.php");?>
<html>
<head>
<title>Aplicación segura</title>
</head>
<body>
<h1>Bienvenido <?php echo $_SESSION["user"];?></h1>
<br/>
----
<br/>
Usuario: <?php echo $_SESSION["user"];?>
<br/>
----
<br/><br/>
<a href="otra.php">Continuar</a>
</body>
</html>
```

otra.php

```
<?php include ("seguridad.php");?>
<html>
<head>
<title>Empresa xyz</title>
</head>
<body>
<h1>Bienvenido <?php echo $_SESSION["user"];?></h1>
<br/><hr/> Sistema de la empresa xyz<hr/><br/>
<br/><br/>
<a href="salir.php">salir</a>
</body>
</html>
```

salir.php

```
<?PHP
session_start(); session_destroy();
?>
<html>
<head> <title>Fin de Sesión</title> </head>
<body>
```

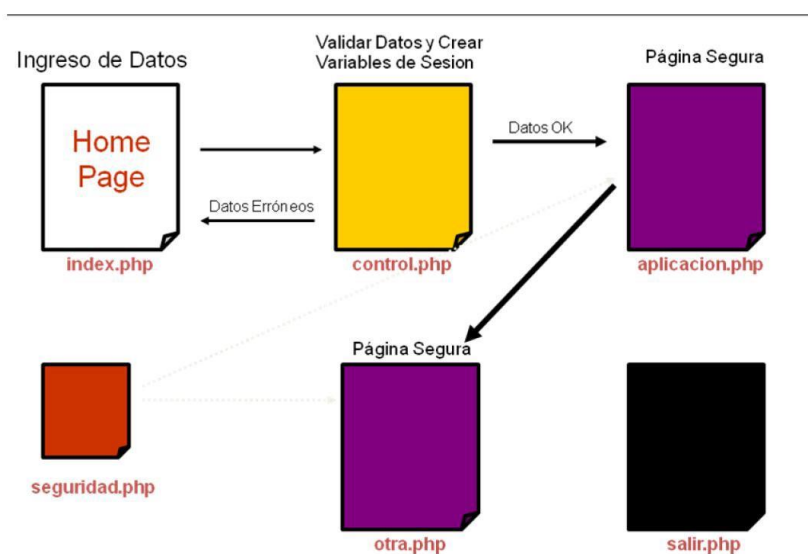
Gracias por tu acceso...

Ir a la Página de Inicio

</body>

</html>

Este mismo ejemplo, de manera gráfica sería:



Todo comienza en el index, pide la autenticación y la envía control.php, esta la verifica, si está OK, pasa a aplicacion.php, caso contrario vuelve a index.php. De aplicación.php podemos ir a otra.php que nuevamente verifica si esta OK, si no es así nos devuelve a index.php y si está OK, nos da la posibilidad de terminar la sesión mediante salir.php

Si quisiéramos ingresar directamente a aplicacion.php o a otra.php nos llevaría a index.php por no estar logueado.

De la misma forma en que lo hemos hecho en el ejemplo anterior el mismo mecanismo se ven en el ejemplo 07_Login el cual incorpora una base de datos contra la cual se compara la clave introducida por el usuario en el formulario de Login.

El ejemplo 08_Registro de usuarios es muy similar al anterior solo que tiene un diseño asociado.

Ejemplo: Recordar contraseña en un equipo

Hay una opción en todos los formularios de acceso a nuestras cuentas de correo, foros, **Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.**

UTN - FRBA. Secretaría de Cultura y Extensión Universitaria

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

blogs, etc. muy famosa por la función que ofrece. Recordar usuario y contraseña en este ordenador es algo que mucha gente utiliza para evitar tener que introducir sus credenciales siempre que acceda a una página web desde su ordenador.

Si bien puede ser un problema importante de seguridad ante una pérdida o robo del equipo, en todos los formularios de acceso podremos ver dicha función.

Para que la página web reconozca siempre nuestro usuario y contraseña en un sitio web, usa las cookies. Ya hemos visto al principio del capítulo que son las cookies, son archivos creados por las páginas web en los que almacena determinada información del visitante, ya sean preferencias o información de su perfil.

Dentro de la denominación de cookie podemos categorizarlas en:

- Cookie de origen: se crean por el dominio del sitio que estamos visitando.
- Cookie de terceros: aunque visitemos un sitio web otros dominios que aparecen en ella pueden crear cookies (por ejemplo los anuncios).

Para crear un sistema que reconozca al usuario incluso al cerrar el navegador y volver a abrirlo vamos a usar un fichero con el formulario y la comprobación de cookies, y otro de autenticación.

En este ejercicio, para hacerlo más sencillo y claro, no me voy a centrar ni en la creación de la sesión de usuario ni en la securización de la identificación.

Antes de nada vamos a crear una base de datos con una tabla de usuarios.

```
CREATE TABLE IF NOT EXISTS `users` (  
  `id_user` int(11) NOT NULL AUTO_INCREMENT,  
  `username` varchar(30) COLLATE utf8_bin NOT NULL,  
  `password` varchar(30) COLLATE utf8_bin NOT NULL,  
  `cookie` varchar(255) COLLATE utf8_bin DEFAULT NULL,  
  PRIMARY KEY (`id_user`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=2 ;
```

La tabla a parte de los datos básicos del usuario llevará un campo llamado cookie donde almacenaremos un valor aleatorio para comprobar la cookie del usuario.

En el archivo index.php, en el formulario pondremos los campos de usuario, contraseña y también el check de recordar contraseña.

```
<form action="auth.php" method="post">  
  <table>  
    <tr>
```

```

        <td>Usuario:</td><td><input type="text" name="username"></td>
    </tr>
    <tr>
        <td>Contrase&ntilde;a:</td><td><input type="text" name="password"></td>
    </tr>
    <tr>
        <td colspan="2"><input type="checkbox" name="recordar"> Recordar
        contrase&ntilde;a</td>
    </tr>
    <tr>
        <td colspan="2"><input type="submit" value="Entrar"></td>
    </tr>
</table>
</form>

```

En el archivo auth.php vamos a realizar la comprobación del usuario y contraseña, a su vez, haremos una comprobación del check de recordar contraseña, y si ha marcado la casilla, crearemos la cookie en el ordenador del visitante por validez de un año.

```

$sql = mysqli_query($conexion, "SELECT * FROM users
                                WHERE username='".$$_POST["username"]."'
                                AND password='".$$_POST["password"]."'"");

if(mysqli_num_rows($sql)==1){
    $row = mysqli_fetch_array($sql);
    $user = $row["username"];
    if(isset($_POST['recordar'])){
        if($_POST['recordar'] == true){
            mt_srand(time());
            $rand = mt_rand(1000000,9999999);
            mysqli_query($conexion, "UPDATE users
                                    SET cookie='".$rand."'
                                    WHERE id_user='".$row["id_user"]")
        }
    }
    or die(mysqli_error($conexion));
    setcookie("id_user", $row["id_user"], time()+(60*60*24*365));
    setcookie("marca", $rand, time()+(60*60*24*365));
    }
    }
    echo "Identificado correctamente.";
}
else{
    echo "Identificaci&oacute;n incorrecta.";
}

```



Posteriormente, en el archivo index.php hacemos una comprobación de nuestra cookie en el ordenador del visitante, y si la encuentra, la compara con el usuario de nuestra base de datos, y con el número aleatorio que tiene asignado.

```
if(isset($_COOKIE['id_user']) && isset($_COOKIE['marca'])){  
    if($_COOKIE['id_user']!=" " || $_COOKIE['marca']!=""){  
        $sql_c = mysqli_query($conexion, "SELECT * FROM users  
            WHERE id_user='".$_COOKIE["id_user"]."  
            AND cookie='".$_COOKIE["marca"]."  
            AND cookie<>";");  
    }  
    if(mysqli_num_rows($sql_c)){  
        $row_c = mysqli_fetch_array($sql_c);  
        echo "El usuario ".$row_c['username']." se ha identificado  
correctamente."  
        $user_cookie = mysqli_fetch_array($sql_c);  
    }  
}
```

Ahora si entramos con el usuario que tenemos en nuestra base de datos y recordamos la contraseña, al cerrar el navegador y volver a entrar en la página nos reconocerá automáticamente.

En la carpeta 09_cookies encontrarán el código completo de este ejemplo y en la carpeta 10_cookies el mismo ejemplo integrado con un html y css.