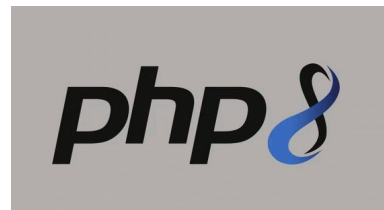


## Aplicacións Web con PHP



## Sumario

Sesións.....	2
<code>session_start()</code> . Inicialización das sesións.....	2
<code>\$_SESSION</code> . Array asociativo.....	3
<code>session_destroy()</code> . Rematando a sesión.....	3
<code>session_unset()</code> . Borrando variables de <code>\$_SESSION</code> .....	4
Autenticación de usuarios e control de acceso.....	5
Autenticación http (WWW-Authenticate).....	5
Distintos usuarios en ficheiro.....	7
Autenticación con <code>htpasswd</code> .....	8
Autenticación con bases de datos.....	9
Cifrado de contrasinais na Base de datos con PHP.....	9

## Sesións

Unha sesión é un bloque de información, gardado no servidor, que almacena todo tipo de variables e valores que garda información sobre os usuarios e as páxinas que visitan no noso sitio web, desde que entran no sitio ata que o abandonan.

A sesión créase no momento que o usuario entre no sitio web: ao crear a sesión asígnaselle un identificador (**Session ID** ou **SID**) en forma de cadea de caracteres, para asociala ao usuario. Este **SID** debe ser propagado de unha a outra páxina no noso sitio web, sempre que o usuario cambia de páxina dentro do noso sitio. Así, na nova páxina pódese recuperar a sesión correspondente.

### ***session\_start( )***. Inicialización das sesións

Para empregar as sesións primeiro hai que iniciar unha sesión. Isto faise coa función `session_start( )`. É moi importante que sexa a primeira liña da páxina .php, para que non produza erros. Se queremos evitar que se mostren warnings ou erros podemos empregar a `@session_start()` (en PHP a arroba antes dunha función evita que se mostren por pantalla warnings e erros).

Exemplo:

```
<?php
@session_start();
echo "Sesión iniciada.";
?>
<!DOCTYPE html>
<html>
...
```

Lembra que ten que ser a primeira liña, sen que existan liñas en branco antes.

Para acceder ao valor do sid, podemos empregar a función `session_id()`:

```
<?php
@session_start();
echo "Sesión iniciada.";
echo " A sesión ten un id de ".session_id();
?>
<!DOCTYPE html>
<html>
...
```

Se temos varios ficheiros enlazados a sesión será a mesma, pois ***session\_start( )*** crea unha sesión nova, ou mantén a que está aberta.

Podemos cambiar entre as dúas páxinas e sempre teremos a mesma sesión.

## **`$_SESSION`. Array asociativo**

Unha vez creada unha sesión, temos dispoñible un array asociativo **`$_SESSION`**, que está dispoñible para o usuario en concreto que iniciou a sesión. Neste array podemos gardar variables que estarán dispoñibles mentres estea aberta a sesión. Por exemplo, podemos dispoñer do nome de usuario ou calquera outra variable en diferentes páxinas enlazadas na mesma sesión:

### **sesion1a.php**

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title></title>
</head>
<body>
<br/>
<!-- DEFINIMOS UNHA VARIABLE -->
<?php
$_SESSION['usuario']="Xan";
?>
<h2>Estou na páxina 1a!! </h2>
<a href="sesion1b.php">Ir a sesion1b </a>

</body>
</html>
```

### **sesion1b.php**

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title></title>
</head>
<body>
<br/>
<?php
/* PODEMOS ACCEDER Á VARIABLE */
echo "O usuario é ",$_SESSION['usuario'];
?>
<h2>Estou na páxina 1b!! </h2>
<a href="sesion1a.php">Ir a sesion1a</a>
<br>
</body>
</html>
```

Deste xeito poderemos manter no array asociativo `$_SESSION` variables que permanecen accesibles mentres teñamos activa a sesión, por exemplo, un usuario, un carrito da compra, etc.

Falta por ver como dar seguridade a estas sesións...

## **`session_destroy()`. Rematando a sesión**

Para rematar a sesión, e que non estea activa, emprégase a sentencia **`session_destroy()`**. Esta non elimina o array asociativo asociado, así que se recomenda borrarlo explicitamente:

```
<?php
session_start( );    /*MANTEMOS A SESIÓN INICIADA */
$_SESSION = array( ); /* ELIMINAMOS TODAS AS VARIABLES */
// E FINALMENTE DESTRUÍMOS A SESIÓN:
session_destroy( );
?>
```

## **`session_unset( )`. Borrando variables de `$_SESSION`**

Outro xeito de borrar é empregar a función `session_unset( )`:

```
<?php
    session_start( );    /*MANTEMOS A SESIÓN INICIADA */
    session_unset() /* ELIMINAMOS TODAS AS VARIABLES */
    // E FINALMENTE DESTRUÍMOS A SESIÓN:
    session_destroy( );
?>
```

Será conveniente ter un ficheiro ***pecharSesion.php*** ou algo parecido para pechar a sesión, que nos leve a login.php:

```
<?php
    session_start( );    /*RETOMAMOS A SESIÓN INICIADA, QUE QUEREMOS PECHAR */
    $_SESSION = array( );

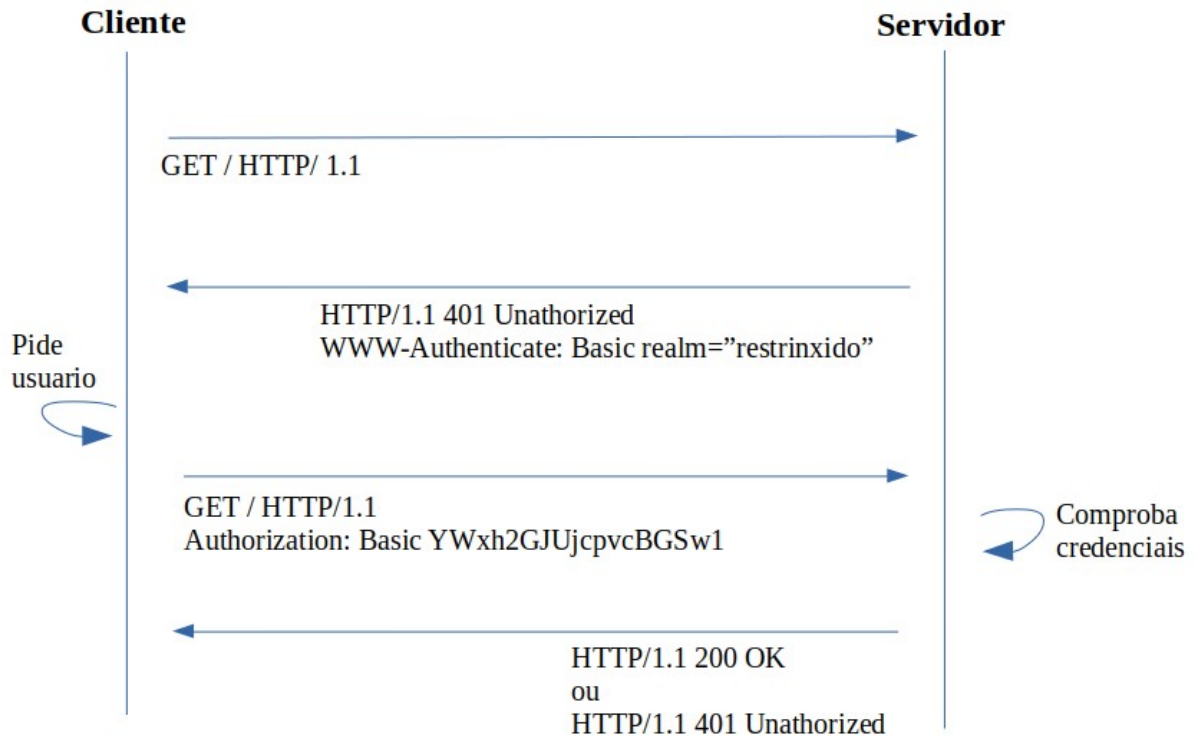
    session_destroy( );

    header("Location:... ");
?>
```

## Autenticación de usuarios e control de acceso

### Autenticación http (WWW-Authenticate)

O protocolo http ofrece un método sinxelo para autenticar os usuarios, baseados no código de estado **401 (Unauthorized)** e no header **WWW-Authenticate**:



O navegador envía unha petición GET ao servidor, e a páxina solicitada ten o encabezamento HTTP WWW-Authenticate. O navegador cando recibe ese encabezamento mostra un formulario de usuario/contrasinal, e estes datos son enviados de volta ao servidor, que comproba credenciais. Se as credenciais son válidas, envíase a páxina solicitada, se non envíase unha mensaxe de erro Non authorized.

En PHP as credenciais grávanse nas variables de servidor:

- ◆ **\$\_SERVER['PHP\_AUTH\_USER']** : o nome de usuario introducido
- ◆ **\$\_SERVER['PHP\_AUTH\_PW']** : contrasinal introducida
- ◆ **\$\_SERVER['AUTH\_TYPE']**: o método http empregado para autenticar. Pode ser "basic" ou "digest".

¿Como temos que escribir en PHP a nosa páxina para que o navegador pida usuario e contrasinal?  
Deste xeito:

//Exemplo1

```
<?php

if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header('WWW-Authenticate: Basic realm="Acceso restrinxido"');
    header('HTTP/1.0 401 Unauthorized');
    echo 'Requerida autenticación para acceder a esta páxina.';
    exit;
}
else {
    echo "<p>Introduciches como nome de usuario: {$_SERVER['PHP_AUTH_USER']}.</p>";
    echo "<p>Introduciches como contrasinal: {$_SERVER['PHP_AUTH_PW']}</p>";
}
```

Tamén poderíamos facer as comprobacións necesarias para determinar se as credenciais son válidas.  
Por exemplo, cun usuario 'proba' e contrasinal 'abc123':

//Exemplo2

```
<?php
if ((!isset($_SERVER['PHP_AUTH_USER'])) ||
    ($_SERVER['PHP_AUTH_USER'] != "proba") || ($_SERVER['PHP_AUTH_PW'] != "abc123"))
{
    header('WWW-Authenticate: Basic realm="Acceso restrinxido"');
    header('HTTP/1.0 401 Unauthorized');
    echo 'Requerida autenticación para acceder a esta páxina.';
    exit;
}
?>
<html>
    <head>
        <title>Exemplo de autenticación http</title>
    </head>
    <body>
        Conseguiu o acceso a zona restrinxida</B>.
    </body>
</html>
```

**OLLO:** Lembra que http envía a información sen cifrar. Para engadir seguridade ao mecanismo de autenticación http temos que traballar SEMPRE con **https/ssl**

## Distintos usuarios en ficheiro

No exemplo anterior un único usuario tiña acceso á parte restrinxida. Nas aplicacións reais precisamos varios usuarios, que podemos gardar nunha base de datos, ou nun ficheiro.

Se empregamos un ficheiro de texto, por exemplo, `passwords.txt`, podemos darlle un formato: `nome_usuario,contrasinal`. Así:

### **passwords.txt**

Ana,abc123.

Xan,123456

Miguel,aabbcc

Agora pedimos a autorización ao comezo da páxina se non estamos xa cun usuario permitido. Comprobamos no ficheiro de contrasinais que chamamos *passwords.txt* se o usuario e contrasinal coinciden, e se é así damos acceso á parte restrinxida.

Para ler o ficheiro empregamos a función *file()* de PHP, que transfere un ficheiro completo a un array, cada fila como un elemento do array `$fich`:

```
<?php
if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header('WWW-Authenticate: Basic realm="Acceso restrinxido"');
    header('HTTP/1.0 401 Unauthorized');
    echo 'Autorizacion requerida.';
    exit;}
else {
    $fich = file("passwords.txt");
    $i = 0;
    $validado = false;
    while (!$validado && $i < count($fich)) {
        $campo = explode(",", $fich[$i]);
        if (($SERVER['PHP_AUTH_USER'] == $campo[0]) && ($SERVER['PHP_AUTH_PW'] ==
            rtrim($campo[1])))
        {
            $validado = true;
        }
        $i++;
    }
    if (!$validado) {
        header('WWW-Authenticate: Basic realm="Acceso restrinxido"');
        header('HTTP/1.0 401 Unauthorized');
        echo 'Autorizacion Requerida.';
        exit;    }
    else {
?>

<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8" />
```

```
<title>Autenticación http</title>
</head>
<body>
    Conseguiu o acceso á zona restrinxida</B> co usuario
    <?php echo $_SERVER['PHP_AUTH_USER'] ?>.
</body>
</html>
<?php
}
}??>
```

## Autenticación con htpasswd

Apache ten unha utilidade en liña de comando **htpasswd**, que permite almacenar un ficheiro diferentes contrasinais, que se almacenarán cifradas. (podes ver <https://httpd.apache.org/docs/2.4/es/howto/auth.html>). O ficheiro é conveniente que estea nunha carpeta non accesible vía web.

Creamos ficheiro **.htpasswd** (podería ser calquera outro nome) na carpeta **/etc/apache2** e engadimos usuario proba con:

```
htpasswd -c /etc/apache2/.htpasswd proba
```

Isto creará o ficheiro (-c), e nos pedirá un password por consola. Poderíamos engadir máis usuarios sen o modificador -c:

```
htpasswd /etc/apache2/.htpasswd xan
```

Agora teremos que indicar no directorio no que teñamos os contidos restrinxidos cales son os usuarios: indicaremos onde está o ficheiro co usuarios e contrasinais. Para isto temos que crear no noso directorio restrinxido un ficheiro **.htaccess** coas directivas de Apache seguintes:

```
AuthName "restrinxindo"
AuthType Basic
AuthUserFile /etc/apache2/.htpasswd
require valid-user
```

Ademais hai que asegurarse de que na configuración de Apache (en apache2.conf) utilízase a directiva **AllowOverwrite** para que se aplique correctamente a configuración dos ficheiros **.htaccess**.

Deste modo Apache só permitirá o acceso á carpeta cos nosos contido restrinxidos, se o usuario e contrasinal está no ficheiro **/etc/apache2/.htpasswd**.



## Autenticación con bases de datos

A autenticación máis avanzada será empregando bases de datos. Os nosos usuarios estarán gardados na base de datos, cun usuario e contrasinal, e para loguearse na nosa aplicación debemos comprobar que son correctos. O contrasinal debe estar cifrado, polo que veremos previamente como é o cifrado de bases de datos

### Cifrado de contrasinais na Base de datos con PHP

Os contrasinais dos usuarios na base de datos deberanse gardar cifrados, para que calquera acceso indebido á mesma non proporcione todos os contrasinais dos nosos usuarios: gardaremos en vez do contrasinal en texto plano o resultado de pasarlle unha función de **hash** a dito contrasinal.

Despois, cando comprobemos o contrasinal tecleado teremos que comprobar que o resultado de pasar a función de hash ao contrasinal tecleado é o mesmo que o gardado na base de datos.

As funcións básicas/antigas de hashing en php eran as seguintes (pasámoslle un string e nos devolve o string hasheado):

- **md5**

```
string md5 (string $str [, bool $raw_output = false ])
```

Exemplo: `$contrasinalHashMd5= md5($contrasinalTecleado);`

Calcula un hash co algoritmo [md5](#). Se se establece `_rawoutput` como true devolverá un raw binario cunha lonxitude de 16. Por defecto un hash de 32 caracteres hexadecimal.

- **sha1**

```
string sha1 (string $str [, bool $raw_output = false ])
```

Calcula un hash con el algoritmo [sha1](#). Se se establece `_rawoutput` como true devolverá un raw binario con una longitud de 20. Por defecto un hash de 40 caracteres hexadecimal.

- **hash**

```
string hash ( string $algoritmo, string $data [, bool $raw_output = false ] )
```

A función toma primeiro o algoritmo que se desexa empregar, *\$algoritmo*, y despois o string que se desexa encriptar, *\$data*. O algoritmo pode ser **md5**, **sha128**, **sha256**... Devolverá o contrasinal encriptado.

A vantaxe de empregar estas funcións é que podemos introducir directamente os contrasinais desde PHPMyAdmin. A desvantaxe é que NON son seguras 100%.

## Librería hash de contrasinais

A extensión hash para contrasinais de PHP crea un password complexo, que se axusta aos estándares de seguridade do momento, polo que é o **método máis recomendado**.

Podemos empregar a función **password\_hash**, co contrasinal que queremos "hashear", e a extensión xa o fai directamente. Recoméndase gardar o resultado nun campo de 255 caracteres.

Empregará o algoritmo que lle indiquemos, se indicamos PASSWORD\_DEFAULT empregará o algoritmo máis forte. Ver <https://www.php.net/manual/es/function.password-hash>

- **password\_hash**

**string password\_hash** (**string \$password**, **integer \$algoritmo** [, **array \$options**])

Por exemplo, podemos "hashear" o noso contrasinal con:

```
$hasheado = password_hash("abc123.", PASSWORD_DEFAULT);
```

Dependendo do número (\$options) o algoritmo será máis complexo e tardará máis en xerarse o hash. Pódese considerar como o número de veces que o algoritmo "hashea" o contrasinal

Para comprobar o contrasinal gardado, agora temos que empregar a función

**boolean password\_verify** (**\$password**, **\$hash**)

Devolverá TRUE se o contrasinal coincide co almacenado, e FALSE en caso contrario.

Exemplo:

```
... XA FIXEMOS A CONSULTA E TEMOS O PASSWORD GARDADO NA BD NA VARIABLE $hash:

if(password_verify($passwordTecleado, $hash)) {
    //PASSWORD CORRECTO    }
else {
    //PASSWORD INCORRECTO  }
```

Empregando esta extensión de cifrado de PHP, a nosa aplicación estará nos últimos estándares de seguridade. A extensión vaise adaptando aos cambios e a nosa aplicación será segura.

Un exemplo podería ser (se temos na nosa BD **proba** unha táboa **usuario** con campos **usuario** e **passwd**):

```
<?php

//SE NON ESTÁ AUTENTICADO PEDIMOS CREDENCIAIS
if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header("WWW-Authenticate: Basic realm='Contido restrinxido'");
    header("HTTP/1.0 401 Unauthorized");
    die();
}
```

```

$host = "db-pdo";
$db = "proba";
$user = "root";
$pass = "root";
$dsn = "mysql:host=$host;dbname=$db;charset=utf8mb4";
try {
    $conPDO = new PDO($dsn, $user, $pass);
    $conPDO->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $ex) {
    die("Erro na conexión mensaxe: " . $ex->getMessage());
}

// COMPROBAMOS SE EXISTE O USUARIO, E RECOLLEMOS O PASSWORD GARDADO NA BD
$consulta = "select passwd from usuario where usuario=:nomeTecleado";
$stmt = $conPDO->prepare($consulta);

try {
    $stmt->execute(array('nomeTecleado' => $_SERVER['PHP_AUTH_USER']));
} catch (PDOException $ex) {
    $conPDO = null;
    die("Erro recuperando os datos da BD: " . $ex->getMessage());
}
$fila=$stmt->fetch();
if($stmt->rowCount() == 1 ) //HAI UN USUARIO
    $contrasinalBD=$fila[0];

$passTecleado=$_SERVER['PHP_AUTH_PW'];

// COMPROBAMOS QUE O HASH GARDADO É COMPATIBLE CO TECLEADO.
//TEMOS QUE COMPROBAR ANTES QUE HAI ALGÚN USUARIO:
if ($stmt->rowCount() == 0 || !password_verify($passTecleado,$contrasinalBD)) {
    header("WWW-Authenticate: Basic realm='Contido restrinxido'");
    header("HTTP/1.0 401 Unauthorized");
    $stmt = null;
    $conProyecto = null;
    die();
}

$stmt = null;
$conPDO = null;
?>
<!DOCTYPE html>
<head>
<title>Autenticación BD</title>
</head>
<body>
<p>
<?php
echo "Benvido {" . $_SERVER['PHP_AUTH_USER'] . "}, está vostede na área restrinxida";
?>
</p>
</body></html>

```