

PHP Cookies

PHP Cookies

- ▶ A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP

A cookie is created with the **setcookie()** function.

Syntax - *setcookie(name, value, expire, path, domain, secure, httponly);*

Only the name parameter is required. All other parameters are optional.

PHP Create/ Retrieve a Cookie

- ▶ The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 * 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).
- ▶ We then retrieve the value of the cookie "user" (using the global variable \$_COOKIE). We also use the isset() function to find out if the cookie is set:
- ▶ Note: The setcookie() function must appear BEFORE the <html> tag.
- ▶ Note: The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use setrawcookie() instead).

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
// 86400 = 1 day
?>

<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

Modify a Cookie Value & Delete a Cookie

To modify a cookie, just set (again) the cookie using the **setcookie()** function:

```
<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>

<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

To delete a cookie, use the **setcookie()** function with an expiration date in the past:

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600);
?>

<html>
<body>

<?php
echo "Cookie 'user' is deleted.";
?>

</body>
</html>
```

Check if Cookies are Enabled

The following example creates a small script that checks whether cookies are enabled. First, try to create a test cookie with the **setcookie()** function, then count the `$_COOKIE` array variable:

```
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
    echo "Cookies are enabled.";
} else {
    echo "Cookies are disabled.";
}
?>

</body>
</html>
```



PHP Sessions

PHP Sessions

An alternative way to make data accessible across the various pages of an entire website is to use a PHP Session.

Unlike a cookie, the information is not stored on the users computer.

A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit.

The location of the temporary file is determined by a setting in the **php.ini** file called **session.save_path**. Before using any session variable make sure you have setup this path.

PHP Sessions

When a session is started following things happen –


- ▶ PHP first creates a unique identifier for that particular session which is a random string of 32 hexadecimal numbers such as 3c7foj34c3jj973hjkop2fc937e3443.
- ▶ A cookie called PHPSESSID is automatically sent to the user's computer to store unique session identification string.
- ▶ A file is automatically created on the server in the designated temporary directory and bears the name of the unique identifier prefixed by sess_ ie sess_3c7foj34c3jj973hjkop2fc937e3443.

When a PHP script wants to retrieve the value from a session variable, PHP automatically gets the unique session identifier string from the PHPSESSID cookie and then looks in its temporary directory for the file bearing that name and a validation can be done by comparing both values.

A session ends when the user loses the browser or after leaving the site, the server will terminate the session after a predetermined period of time, commonly 30 minutes duration.

Starting a PHP Session

- ▶ A PHP session is easily started by making a call to the **session_start()** function. This function first checks if a session is already started and if none is started then it starts one. It is recommended to put the call to `session_start()` at the beginning of the page.
- ▶ Session variables are stored in associative array called `$_SESSION[]`. These variables can be accessed during lifetime of a session.
- ▶ The following example starts a session then register a variable called counter that is incremented each time the page is visited during the session.
- ▶ Make use of `isset()` function to check if session variable is already set or not.
- ▶ Put this code in a `test.php` file and load this file many times to see the result –



```
<?php
    session_start();

    if( isset( $_SESSION['counter'] ) ) {
        $_SESSION['counter'] += 1;
    }else {
        $_SESSION['counter'] = 1;
    }

    $msg = "You have visited this page ". $_SESSION['counter'];
    $msg .= "in this session.";
?>

<html>

    <head>
        <title>Setting up a PHP session</title>
    </head>

    <body>
        <?php echo ( $msg ); ?>
    </body>

</html>
```

It will produce the following result –

You have visited this page 1 in this session.

Turning on Auto Session



You don't need to call `start_session()` function to start a session when a user visits your site if you can set **`session.auto_start`** variable to 1 in **`php.ini`** file.

Sessions without cookies

There may be a case when a user does not allow to store cookies on their machine. So there is another method to send *session ID* to the browser.

Alternatively, you can use the constant `SID` which is defined if the session started. If the client did not send an appropriate session cookie, it has the form **`session_name=session_id`**. Otherwise, it expands to an empty string. Thus, you can embed it unconditionally into URLs.

The following example demonstrates how to register a variable, and how to link correctly to another page using **`SID`**.

Destroying a PHP Session

- ▶ A PHP session can be destroyed by **session_destroy()** function. This function does not need any argument and a single call can destroy all the session variables. If you want to destroy a single session variable then you can use unset() function to unset a session variable.
- ▶ Here is the example to unset a single variable –

```
<?php  
    unset($_SESSION['counter']);  
?>
```

Here is the call which will destroy all the session variables –

```
<?php  
    session_destroy();  
?>
```



Thank You

Vidyashankara