php sessions - why use them?

As a website becomes more sophisticated, so must the code that backs it. When you get to a stage where your website need to pass along user data from one page to another, it might be time to start thinking about using PHP sessions.

A normal HTML website will not pass data from one page to another. In other words, all information is forgotten when a new page is loaded. This makes it quite a problem for tasks like a shopping cart, which requires data (the user's selected product) to be remembered from one page to the next.

php sessions - overview

A PHP session solves this problem by allowing you to store user information on the server for later use (i.e. username, shopping cart items, etc). However, this session information is temporary and is usually deleted very quickly after the user has **left** the website that uses sessions.

It is important to ponder if the sessions' temporary storage is applicable to your website. If you require a more permanent storage you will need to find another solution, like a MySQL database.

Sessions work by creating a unique identification(UID) number for each visitor and storing variables based on this ID. This helps to prevent two users' data from getting confused with one another when visiting the same webpage.

Note:If you are not experienced with session programming it is not recommended that you use sessions on a website that requires high-security, as there are security holes that take some advanced techniques to plug.

starting a php session

Before you can begin storing user information in your PHP session, you must first start the session. When you start a session, it must be at the very beginning of your code, before any HTML or text is sent.

Below is a simple script that you should place at the beginning of your

PHP code to start up a PHP session.

PHP Code:

```
<?php
session_start(); // start up your PHP session!
?>
```

This tiny piece of code will register the user's session with the server, allow you to start saving user information and assign a UID (unique identification number) for that user's session.

storing a session variable

When you want to store user data in a session use the $_SESSION associative array. This is where you both store and retrieve session data. In previous versions of PHP there were other ways to perform this store operation, but it has been updated and this is the correct way to do it.

PHP Code:

```
<?php
session_start();
$_SESSION['views'] = 1; // store session data
echo "Pageviews = ". $_SESSION['views']; //retrieve data
?>
```

Display:

```
Pageviews = 1
```

In this example we learned how to store a variable to the session associative array $_SESSION and also how to retrieve data from that same array.

php sessions: using php's *isset* function

Now that you are able to store and retrieve data from the $_SESSION array, we can explore some of the real functionality of sessions. When you create a variable and store it in a session, you probably want to use it in the future. However, before you use a session variable it is necessary that you check to see if it exists already!

This is where PHP's *isset* function comes in handy. *isset* is a function that takes any variable you want to use and checks to see if it has been **set**. That is, it has already been assigned a value.

With our previous example, we can create a very simple pageview counter by using *isset* to check if the pageview variable has already been created. If it has we can increment our counter. If it doesn't exist we can create a pageview counter and set it to one. Here is the code to get this job

done:

PHP Code:

```
<?php
session_start();
if(isset($_SESSION['views']))
   $_SESSION['views'] = $_SESSION['views']+ 1;
else
   $_SESSION['views'] = 1;

echo "views = ". $_SESSION['views'];
?>
```

The first time you run this script on a **freshly opened browser** the *if statement* will fail because no session variable *views* would have been stored yet. However, if you were to refresh the page the *if statement* would be true and the counter would increment by one. Each time you reran this script you would see an increase in *view* by one.

cleaning and destroying your session

Although a session's data is temporary and does not require that you explicitly clean after yourself, you may wish to delete some data for your various tasks.

Imagine that you were running an online business and a user used your website to buy your goods. The user has just completed a transaction on your website and you now want to remove everything from their shopping cart.

PHP Code:

```
<?php
session_start();
if(isset($_SESSION['cart']))
   unset($_SESSION['cart']);
?>
```

You can also completely destroy the session entirely by calling the *session_destroy* function.

PHP Code:

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

Destroy will reset your session, so don't call that function unless you are entirely comfortable losing all your stored session data!

Ads by Google	PHP	PHP and MySQL	Script En PHP	Array in PHP