جامعة الانبار

كلية علوم الحاسوب وتكنولوجيا المعلومات

قسم أنظمة شبكات الحاسوب

برمجة صفحات الويب PHP

المرحلة الثالثة

الفصل الدراسي الاول

مدرس المادة

م.د. سميه عبدالله حمد

Lectures:

١. lec١: PHP Fundamentals (اساسيات برمجة صفحات الانترنت)
٢. lec٢: PHP Data Types (أنواع البيانات)
٣. lec٣: PHP Arrays (المصفوفات)
٤. lec٤: PHP Logic Control Structures (عبارات السيطرة)
٥. lec٥: PHP Loops (عبارات التكرار)
٦. lec٦: PHP Strings PHP String Functions (السلاسل ودوالها)
٧. lec٧: PHP Function (الدوال)
٨. lec٨: PHP Date() & Time Function (دوال الوقت والتاريخ)
٩. lec٩: Regular Expressions (التعابير المنتظمة)
١٠. lec١٠: PHP File Handling & Functions (دوال وتحميل الملفات)
١١. lec١١: PHP Session & PHP Cookies (الجلسات وملفات تعريف الارتباط)
١٢. lec١٢: How to Send Email using PHP mail (كيفية ارسال الايميل)

7- PHP Session & PHP Cookies with Example

What is Cookie?

A cookie is a small file with the maximum size of 4KB that the web server stores on the client computer.

Once a cookie has been set, all page requests that follow return the cookie name and value.

A cookie can only be read from the domain that it has been issued from. For example, a cookie set using the domain www.guru99.com can not be read from the domain career.guru99.com.

Most of the websites on the internet display elements from other domains such as advertising. The domains serving these elements can also set their own cookies. These are known as third party cookies.

A cookie created by a user can only be visible to them. Other users cannot see its value.

Most web browsers have options for disabling cookies, third party cookies or both.

If this is the case then PHP responds by passing the cookie token in the URL.

The diagram shown below illustrates how cookies work.

Here,

1) A user requests for a page that stores cookies

2) The server sets the cookie on the user's computer

3) Other page requests from the user will return the cookie name and value

Why and when to use Cookies?

- Http is a stateless protocol; cookies allow us to track the state of the application using small files stored on the user's computer.The path were the cookies are stored depends on the browser.Internet Explorer usually stores them in Temporal Internet Files folder.
- Personalizing the user experience – this is achieved by allowing users to select their preferences.The page requested that follow are personalized based on the set preferences in the cookies.
- Tracking the pages visited by a user

Creating Cookies

Let's now look at the basic syntax used to create a cookie.

```
<?php

setcookie(cookie_name, cookie_value, [expiry_time], [cookie_path], [domain], [secure], [httponly]);

?>
```

HERE,

- Php"setcookie" is the PHP function used to create the cookie.
- "cookie_name" is the name of the cookie that the server will use when retrieving its value from the $_COOKIE array variable. It's mandatory.
- "cookie_value" is the value of the cookie and its mandatory
- "[expiry_time]" is optional; it can be used to set the expiry time for the cookie such as 1 hour. The time is set using the PHP time() functions plus or minus a number of seconds greater than 0 i.e. time() + 3600 for 1 hour.
- "[cookie_path]" is optional; it can be used to set the cookie path on the server. The forward slash "/" means that the cookie will be made

available on the entire domain. Sub directories limit the cookie access to the subdomain.

- "[domain]" is optional, it can be used to define the cookie access hierarchy i.e. www.cookiedomain.com means entire domain while www.sub.cookiedomain.com limits the cookie access to www.sub.cookiedomain.com and its sub domains. *Note it's possible to have a subdomain of a subdomain as long as the total characters do not exceed 253 characters.*
- "[secure]" is optional, the default is false. It is used to determine whether the cookie is sent via https if it is set to true or http if it is set to false.
- "[Httponly]" is optional. If it is set to true, then only client side scripting languages i.e. JavaScript cannot access them.

Note: the php set cookie function must be executed before the HTML opening tag.

Let's now look at an example that uses cookies.

We will create a basic program that allows us to store the user name in a cookie that expires after ten seconds.

The code below shows the implementation of the above example "cookies.php".

```
<?php
    setcookie("user_name", "Guru99", time()+ 60,'/'); // expires after 60 seconds
    echo 'the cookie has been set for 60 seconds';
?>
```

Output:

the cookie has been set for 60 seconds

Retrieving the Cookie value

Create another file named "cookies_read.php" with the following code.

```
<?php
    print_r($_COOKIE);    //output the contents of the cookie array variable
?>
```

Output:

Array ([PHPSESSID] => h5onbf7pctbr0t68adugdp2611 [user_name] => Guru99)

Note: $_COOKIE is a PHP built in super global variable.

It contains the names and values of all the set cookies.

The number of values that the

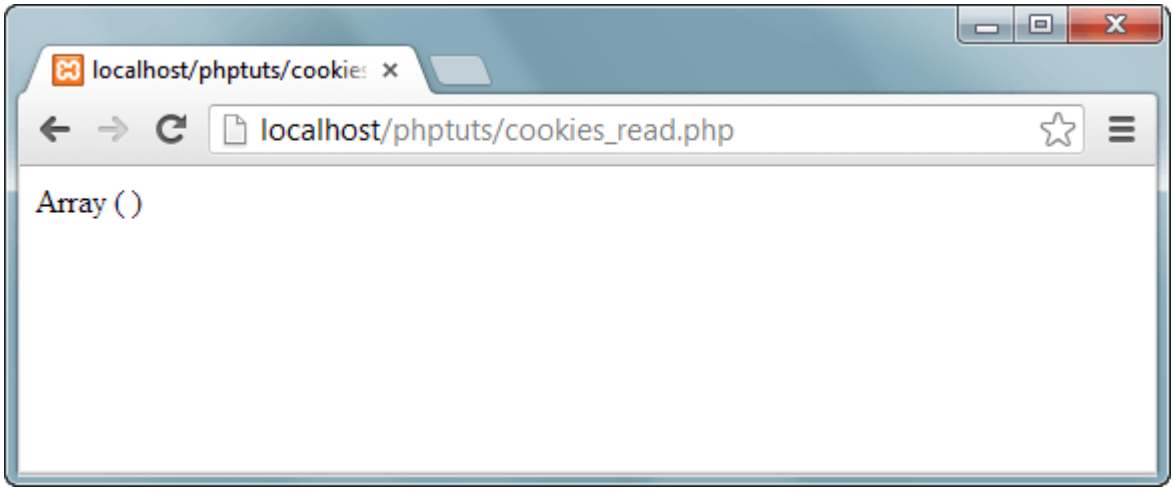
$_COOKIE array can contain depends on the memory size set in php.ini.

The default value is 1GB.

Testing our application.

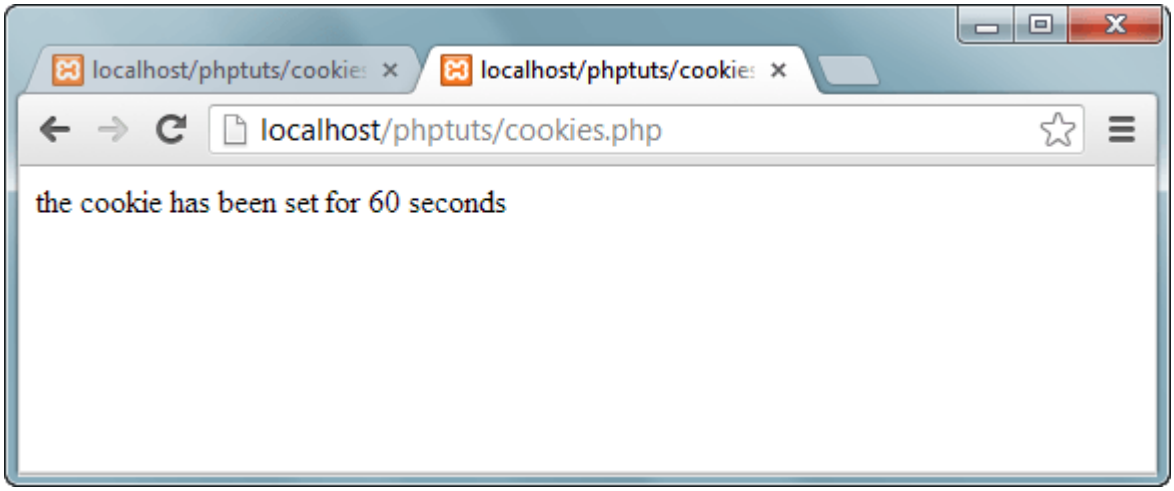
Let's assume you have saved your PHP files in phptus folder.

- Step 1 – open your web browser and enter the URL **http://localhost/phptuts/cookies_read.php**



Note: Only an empty array has been displayed

- Step 2 – Browser to the URL **http://localhost/phptuts/cookies.php**



- Step 3 – Switch back to the first tab then click on refresh button

Wait for a minute then click on refresh button again. What results did you get?

Delete Cookies

- If you want to destroy a cookie before its expiry time, then you set the expiry time to a time that has already passed.
- Create a new filed named cookie_destroy.php with the following code

```
<?php

setcookie("user_name", "Guru99", time() - 360,'/');
```

```
?>
```

- Repeat steps 1 through to 3 from the above section on retrieving cookie values.
- Open the URL **http://localhost/phptuts/cookie_destroy.php**
- Switch to the URL **http://localhost/phptuts/cookies_read.php** what results does it display?

What is a Session?

- A session is a global variable stored on the server.
- Each session is assigned a unique id which is used to retrieve stored values.
- Whenever a session is created, a cookie containing the unique session id is stored on the user's computer and returned with every request to the server. If the client browser does not support cookies, the unique php session id is displayed in the URL
- Sessions have the capacity to store relatively large data compared to cookies.
- The session values are automatically deleted when the browser is closed. If you want to store the values permanently, then you should store them in the database.
- Just like the $_COOKIE array variable, session variables are stored in the $_SESSION array variable. Just like cookies, the session must be started before any HTML tags.

Why and when to use Sessions?

- You want to store important information such as the user id more securely on the server where malicious users cannot temper with them.
- You want to pass values from one page to another.
- You want the alternative to cookies on browsers that do not support cookies.
- You want to store global variables in an efficient and more secure way compared to passing them in the URL
- You are developing an application such as a shopping cart that has to temporary store information with a capacity larger than 4KB.

Creating a Session

In order to create a session, you must first call the PHP session_start function and then store your values in the $_SESSION array variable.

Let's suppose we want to know the number of times that a page has been loaded, we can use a session to do that.

The code below shows how to create and retrieve values from sessions

```
<?php

session_start(); //start the PHP_session function

if(isset($_SESSION['page_count']))
{
    $_SESSION['page_count'] += 1;
}
else
{
    $_SESSION['page_count'] = 1;
}
 echo 'You are visitor number ' . $_SESSION['page_count'];

?>
```

Output:

You are visitor number 1

Destroying Session Variables

The session_destroy() function is used to destroy the whole Php session variables.

If you want to destroy only a session single item, you use the unset() function.

The code below illustrates how to use both methods.

```
<?php

 session_destroy(); //destroy entire session

?>
```

```
<?php
unset($_SESSION['product']); //destroy product session item

?>
```

Session_destroy removes all the session data including cookies associated with the session.

Unset only frees the individual session variables.

Other data remains intact.

Summary

- Cookies are small files saved on the user's computer
- Cookies can only be read from the issuing domain
- Cookies can have an expiry time, if it is not set, then the cookie expires when the browser is closed
- Sessions are like global variables stored on the server
- Each session is given a unique identification id that is used to track the variables for a user.
- Both cookies and sessions must be started before any HTML tags have been sent to the browser.