PHP Cookies & Session

What are cookies?

Cookies are small pieces of information (nearly 4KB) that a website stores on on the user's computer either in the memory of user's computer or other devices such as mobile phones, tablet devices (these cookies are commonly known as session cookies) or are placed on the hard drive of your device (commonly known as persistent cookies).

Cookies can be created when you visit a website or other service that uses cookies. Most of the time these cookies contain harmless non-identifiable (anonymous) information that help the website provide a better user experience to you; or to assist in improving the website by tracking your interactions.

Why we use cookies

There are many reasons for a website to use cookies. Anything from allowing you to login to a secure area of a website, to remembering your name or favorite color for next time you visit the site. The cookies we use only contain anonymous information to improve the services we offer, and do not contain personal information.

Setting a Cookie in PHP

The setcookie() function is used to set a cookie in PHP. Make sure you call the setcookie() function before any output generated by your script otherwise cookie will not set.

The basic syntax of this function can be given with:

```
setcookie(name, value, expire, path, domain, secure);
```

hafizur.kl@diit.info

01739-981172

The parameters of the setcookie() function has the following meaning:

Parameter	Description
name	The name of the cookie.
value	The value of the cookie. Do not store sensitive information since this value is stored on the user's computer.
expires	The expiry date in UNIX timestamp format. After this time cookie will become inaccessible. The default value is 0.
path	Specify the path on the server for which the cookie will be available. If set to'/', the cookie will be available within the entire domain.
domain	Specify the domain for which the cookie is available to e.g www.example.com.
secure	This field, if present, indicates that the cookie should be sent only if a secure HTTPS connection exists.

Here's an example that uses setcookie() function to create a cookie named "username"and assign the value value "John Carter" to it. It also specify that the cookie will expire after 30 days (30 days * 24 hours * 60 min * 60 sec).

```
<?php
// Setting a cookie
setcookie("username", "Hafizur Rahman", time()+30*24*60*60);
?>
```

Accessing Cookies Values

The PHP $_COOKIE superglobal variable is used to retrieve a cookie value. It typically an associative array that contains a list of all the cookies values sent by the browser in the current

request, keyed by cookie name. The individual cookie value can be accessed using standard array notation, for example to display the username cookie set in the previous example, you could use the following code.

```
<?php
// Accessing an individual cookie value
echo $_COOKIE["username"];
?>
```

The PHP code in the above example produce the following output.

Hafizur Rahman

It's a good practice to check whether a cookie is set or not before accessing its value. To do this you can use the PHP isset() function, like this:

```
<?php
// Verifying whether a cookie is set or not
if(isset($_COOKIE["username"])){
    echo "Hi " . $_COOKIE["username"];
} else{
    echo "Welcome Guest!";
}
?>
```

Removing Cookies

You can delete a cookie by calling the same setcookie() function with the cookie name and any value (such as an empty string) however this time you need the set the expiration date in the past, as shown in the example below:

PHP - Sessions

A PHP session is used to store certain data on the server on a temporary basis.

What is a PHP Session?

When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.

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.

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.

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.

hafizur.kl@diit.info

01739-981172

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>
```

OR

```
<?php
// Starting session
session_start();

// Storing session data
$_SESSION["firstname"] = "Hafizur";
$_SESSION["lastname"] = "Rahman";

/ Accessing session data
echo 'Hi, ' . $_SESSION["firstname"] . ' ' . $_SESSION["lastname"];
?>
```

Destroying a Session

If you want to remove certain session data, simply unset the corresponding key of the $_SESSION associative array, like this.

```
<?php
// Starting session
session_start();

// Removing session data
if(isset($_SESSION["lastname"])){
    unset($_SESSION["lastname"]);
}
?>
```

However to destroy a session completely, simply call the session_destroy() *function. This function does not need any argument and a single call destroys all the session data.*

```
<?php
// Starting session
session_start();

// Destroying session
session_destroy();
?>
```

hafizur.kl@diit.info
01739-981172