Unit – 4

Cookies and Sessions

Cookie

- 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.

Cookie

- Cookies are **small files** which are stored on a user's computer.
- They are designed to hold a **modest amount of data specific to a particular client and website**, and can be accessed either by the web server or the client computer.
- This allows the server to **deliver a page tailored to a particular user,** or the page itself can contain some script which is aware of the data in the cookie and so is able to carry information from one visit to the website (or related site) to the next.

- Each cookie is effectively a **small lookup table containing pairs of (key, data**) values - for example (firstname, John) (lastname, Smith). Once the cookie has been read by the code on the server or client computer, the data can be retrieved and used to customised the web page appropriately.

- Writing data to a cookie is usually done when a new webpage is loaded - for example after a 'submit' button is pressed the data handling page would be responsible for storing the values in a cookie.

- If the user has **elected to disable cookies then the write operation will fail**, and subsequent sites which rely on the cookie will either have to take a default action, or prompt the user to re-enter the information that would have been stored in the cookie.

Why are Cookies Used?

- Cookies are a convenient way to carry information from one session on a website to another, or between sessions on related websites, without having to burden a server machine with massive amounts of data storage.

- Storing the data on the server without using cookies would also be problematic because it would be difficult to retrieve a particular user's information without requiring a login on each visit to the website.

- If there is a **large amount of information to store, then a cookie can simply be used as a means to identify a given user** so that further related information can be looked up on a server-side database.

- For example the first time a user visits a site they may choose a username which is stored in the cookie, and then provide data such as password, name, address, preferred font size, page layout, etc. - this information would all be stored on the database using the username as a key.

- Subsequently when the site is revisited the server will read the cookie to find the username, and then retrieve all the user's information from the database without it having to be re-entered.

- The time of expiry of a cookie can be set when the cookie is created. By default the cookie is destroyed when the current browser window is closed, but it can be made to persist for an arbitrary length of time after that

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.

Parameter	Description
name	Required. Specifies the name of the cookie
value	Optional. Specifies the value of the cookie
expire	Optional. Specifies when the cookie expires. The value: time()+86400*30, will set the cookie to expire in 30 days. If this parameter is omitted or set to 0, the cookie will expire at the end of the session (when the browser closes). Default is 0
path	Optional. Specifies the server path of the cookie. If set to "/", the cookie will be available within the entire domain. If set to "/php/", the cookie will only be available within the php directory and all sub-directories of php. The default value is the current directory that the cookie is being set in
domain	Optional. Specifies **the domain name of the cookie**. To make the cookie available on all subdomains of example.com, set domain to "example.com". Setting it to www.example.com will make the cookie only available in the www subdomain
secure	Optional. Specifies **whether or not the cookie should only be transmitted over a secure HTTPS connection.** TRUE indicates that the cookie will only be set if a secure connection exists. Default is FALSE
httponly	Optional. **If set to TRUE the cookie will be accessible only through the HTTP protocol** (the cookie will not be accessible by scripting languages). This setting can help to reduce identity theft through XSS attacks. Default is FALSE

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.

Example

```
<!DOCTYPE html>
<?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];
}
?>

<p><strong>Note:</strong> You might have to reload the page to see the value of the cookie.</p>

</body>
</html>
```

- **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).**

Modify a Cookie Value

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

```
<!DOCTYPE html>
<?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];
}
?>

<p><strong>Note:</strong> You might have to reload the page to see the new value of the
cookie.</p>

</body>
</html>
```

Delete a Cookie

- To delete a cookie, use the **setcookie() function with an expiration date in the past:**
- <!DOCTYPE html>
  <?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:
- <!DOCTYPE html>
<?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>

SESSION

Session

- A session is a way to store information (in variables) to be used across multiple pages.

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

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.

- So, Session variables hold information about one single user, and are available to all pages in one application.

Difference between Session and Cookies?

- **Cookie:**
  - A cookie keep information in the user's browser until it is deleted from there.
  - We can store almost anything in browser cookies.
  - The problem with cookies are that anyone can block or delete cookie content at any time, so if any application is dependent on browser cookie then that application would not work if cookie is deleted or blocked.
- **Session:**
  - Sessions are implemented by using cookies, the actual data is not in the browser; it is stored in the user's session record on the server.
  - They work instead like a token allowing access and passing information while the user has their browser open.
  - The main problem with session is that as soon as we close the browser we lose the session and all the information get lost.

Key differences

Cookie	Session
Cookies are stored in user's browser.	Sessions are not stored in user's browser.
Cookie data are stored at client side.	Session data are stored at server side.
Cookie data are easy to modify as they are stored at client side.	Session data are not easy to modify as they are stored at server side.
Cookie data is available in our browser up to expiration date.	Session data is available for the browser run. After closing the browser we lose the session information.
Cookies saves local user data as website specific on local computer on text file.	Session is application specific and keeps information until the browser is open.

More Specifically

- A cookie is just a key-value pair that is stored in the user's browser. A cookie is sent to your browser as part of the HTTP response that contains the web page you requested.
- When your browser receives a cookie, it stores it, and sends it back to the server with every subsequent request it makes on the same website.
- Because cookies are part of the HTTP request and response headers, they are somewhat limited in size.
- They can be used to store user's preferences, such as the preferred language or currency, or unique tracking IDs for statistical purposes (helps to identify a returning customer), or session IDs.
- **For larger, or sensitive data, you typically store values in the session. The cookie is only there to identify the proper session.**
- A cookie can be configured to only live until the browser window is closed, or have a configurable lifetime (1 week, 1 month, 1 year, whatever). If you visit the website again during this period, your browser will send the cookie with every request.

- A session is a set of data that is stored on the server, usually as **key-value pairs.**
- A session is assigned a **pseudo-random, secret ID** that is usually stored in the user's browser using a cookie, for example SESSID=abcdef123456789.
- The session ID typically matches the name of a file containing the session data on the server.
- Sessions are usually short-lived, and automatically deleted if unused for some time (20 minutes or so).

Start a PHP Session

- A session is started with the **session_start()** function.
- Session variables are set with the PHP global variable: $_SESSION.
- Now, let's create a new page called a"demo_session1.php". In this page, we start a new PHP session and set some session variables:

Example

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```

- **Note: The session_start() function must be the very first thing in your document before any HTML tags.**

Get PHP Session Variable Values

- Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php").

- Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session_start()).

- Also notice that all session variable values are stored in the global $_SESSION variable:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>

</body>
</html>
```

Example

- Another way to show all the session variable values for a user session is to run the following code:
- <?php
  session_start();
  ?>
  <!DOCTYPE html>
  <html>
  <body>

  <?php
  print_r($_SESSION);
  ?>

  </body>
  </html>

How does it work? How does it know it's me?

- Most sessions set a user-key on the user's computer that looks something like this: 765487cf34ert8dede5a562e4f3a7e12.

- Then, when a session is opened on another page, it scans the computer for a user-key. If there is a match, it accesses that session, if not, it starts a new session.

Modify a PHP Session Variable

- To change a session variable, just overwrite it:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// to change a session variable, just overwrite it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>

</body>
</html>
```

Destroy a PHP Session

- To remove all global session variables and destroy the session, use session_unset() and session_destroy():
- <?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// remove all session variables
session_unset();

// destroy the session
session_destroy();

echo "All session variables are now removed, and the session is destroyed."
?>

</body>
</html>