PHP Programming Basic Skills (2)

Wen-Hsiang Lu (盧文祥)
Department of Computer Science and Information Engineering,
National Cheng Kung University
2014/11/18

Outline

- File Manipulation
- File Management
- Interaction Tracking
  - Session & cookie

File Manipulation

- A file manipulation session might involve the following steps:
  1. Open the file for read/write.
  2. Read in the file.
  3. Close the file (may happen later).
  4. Perform operations on the file contents.
  5. Write results out.

```
$fp = fopen($filename, “r+”) or die(“Can’t open file $filename”);
$fstring = fread($fp, filesize($filename));
$fout = fwrite($fp, $fstring);
fclose($fp);
```

File Open

- fopen(): return a string Resource id #n,
- A file is opened in six modes:

✦ Read-only (“r”).

✦ Read and write if the file exists already (“r+”): will write to the beginning of the file, doubling original contents of the file if you read the file in as a string, edit it, and then write the string out to the file.

✦ Write-only (“w”) will create a file of this name, if one doesn’t already exist, and will erase the contents of any file of this name before writing! You cannot use this mode to read a file, only to write one.

✦ Write and read even if the file doesn’t exist already (“w+”) : will create a file of this name, if one doesn’t already exist, and will erase the contents of any file of this name before writing!

✦ Write-only to the end of a file whether it exists or not (“a”).

✦ Read and write to the end of a file whether it exists or not (“a+”), “doubling” original contents of the file if you read the file in as a string, edit it, and then write the string out to the file.

File Open

- HTTP fopen
  - $fp = fopen(“http://www.example.com/ofile.html/”, “r”);
- FTP fopen
  - $fp = fopen(“ftp://username:password@example.com/ofile.txt/”, “r”);
- Window system fopen
  - $fp = fopen("c:\\xampp\\htdocs\\basic.php","r")

File Read

- fread(): a whole file reading
  - $fstring = fread($fp, filesize($filename));
  - **[Ex]**php2-0-file-open.php
- fgets(): line-by-line file reading
  - $line_string= fgets($fp, $length);
  - **[Ex]**php2-1-file-open.php
- file_get_contents()
  - $url = "http://www.google.com/";
  - $contents = file_get_contents($url);
  - **[Ex]**php1-5-getHtml.php

Application: Meta Search

- Meta-search engine: dispatch the user query to several engines at same time, collect and merge the results into one list to the user.
- Extra bonus: Develop a meta-search engine which responds user queries with combined search results from a few search engines.
- Tips: use file_get_contents()
  - $url = "http://www.google.com/";
  - $contents = file_get_contents($url);
  - **[Ex]** **php1-5-getHtml.php**

File Write & Close

- fwrite() & fputs()

```
$fout = fwrite($fp, $fstring);
if ($fout  !=  strlen($fstring)){
    echo “file write failed!”;
}
```

  - **[Ex]**[php2-2-file-write.php]

Common Filesystem Function

- feof(): tests for end-of-file on a file

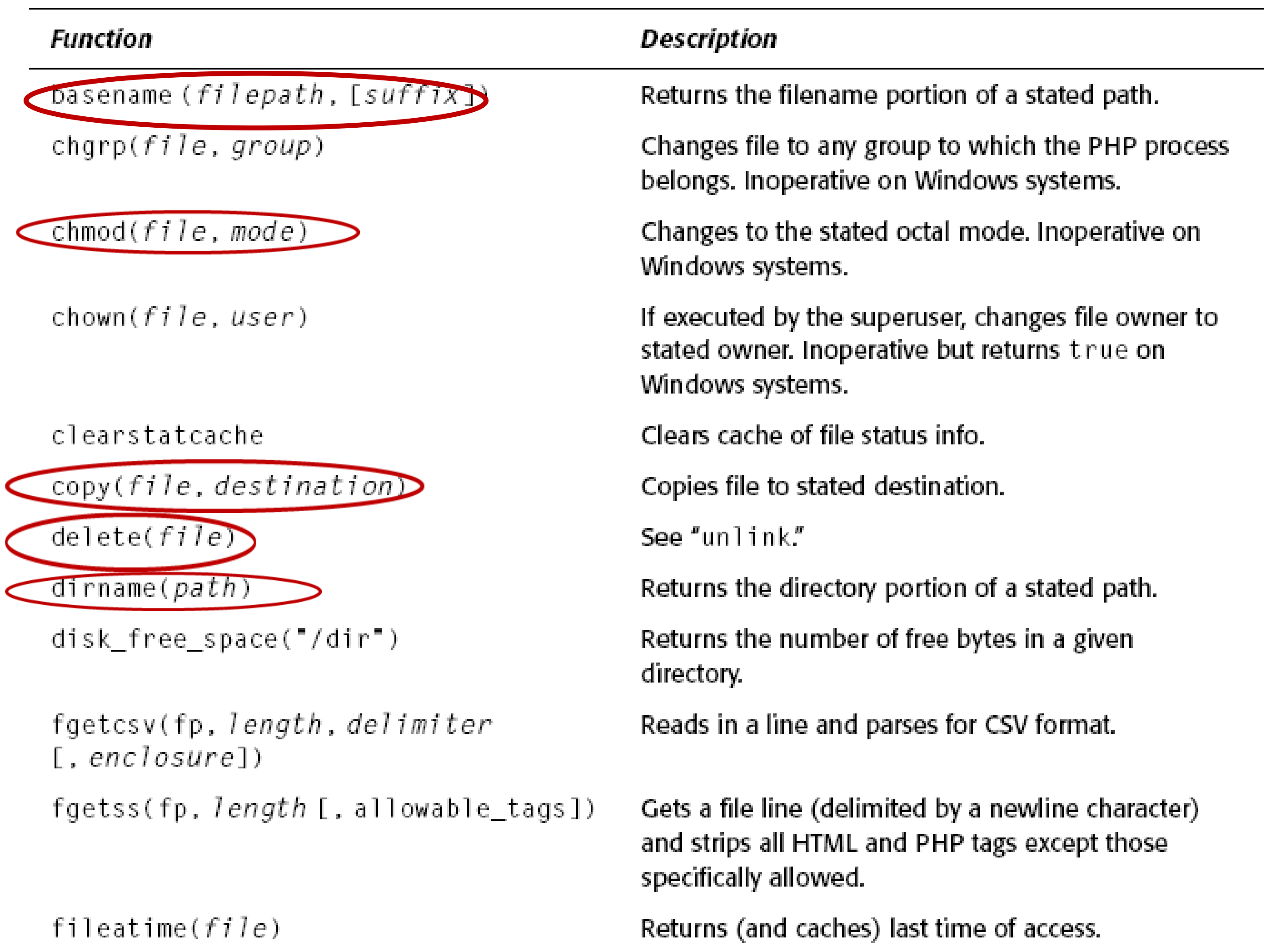
```
while (!feof($fp)) {
  $line = fgets($fp, 4096);
  echo $line;
}
```

- file_exists : checks whether a file exists

```
if (!file_exists(“testfile.php”)) {
    $fp = fopen(“testfile.php”, “w+”);
}
```

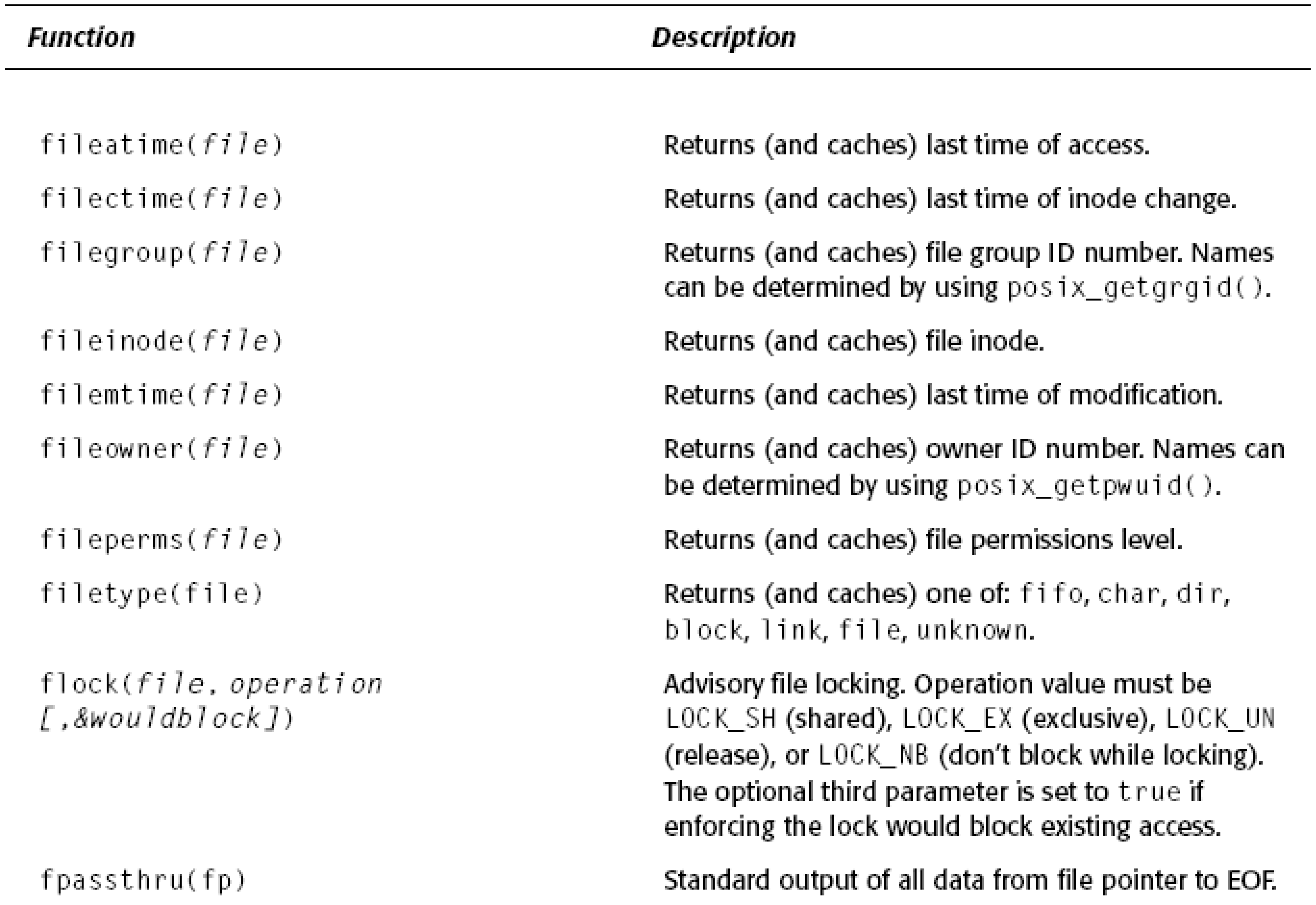
- filesize: returns the size of a file in bytes.
  - $fstring= fread($fp, filesize($filename));

Common Filesystem Function



Function	*Description*
`basename(filepath, [suffix])`	Returns the filename portion of a stated path.
`chgrp(file, group)`	Changes file to any group to which the PHP process belongs. Inoperative on Windows systems.
`chmod(file, mode)`	Changes to the stated octal mode. Inoperative on Windows systems.
`chown(file, user)`	If executed by the superuser, changes file owner to stated owner. Inoperative but returns `true` on Windows systems.
`clearstatcache`	Clears cache of file status info.
`copy(file, destination)`	Copies file to stated destination.
`delete(file)`	See "`unlink`."
`dirname(path)`	Returns the directory portion of a stated path.
`disk_free_space("/dir")`	Returns the number of free bytes in a given directory.
`fgetcsv(fp, length, delimiter [, enclosure])`	Reads in a line and parses for CSV format.
`fgetss(fp, length [, allowable_tags])`	Gets a file line (delimited by a newline character) and strips all HTML and PHP tags except those specifically allowed.
`fileatime(file)`	Returns (and caches) last time of access.

Common Filesystem Function



Function	*Description*
`fileatime(`*`file`*`)`	Returns (and caches) last time of access.
`filectime(`*`file`*`)`	Returns (and caches) last time of inode change.
`filegroup(`*`file`*`)`	Returns (and caches) file group ID number. Names can be determined by using `posix_getgrgid()`.
`fileinode(`*`file`*`)`	Returns (and caches) file inode.
`filemtime(`*`file`*`)`	Returns (and caches) last time of modification.
`fileowner(`*`file`*`)`	Returns (and caches) owner ID number. Names can be determined by using `posix_getpwuid()`.
`fileperms(`*`file`*`)`	Returns (and caches) file permissions level.
`filetype(file)`	Returns (and caches) one of: `fifo`, `char`, `dir`, `block`, `link`, `file`, `unknown`.
`flock(`*`file, operation [,&wouldblock]`*`)`	Advisory file locking. Operation value must be `LOCK_SH` (shared), `LOCK_EX` (exclusive), `LOCK_UN` (release), or `LOCK_NB` (don't block while locking). The optional third parameter is set to `true` if enforcing the lock would block existing access.
`fpassthru(fp)`	Standard output of all data from file pointer to EOF.

Common Filesystem Function

Function	***Description***
`fseek(fp, offset, whence)`	Moves file pointer offset number of bytes into file stream from the position indicated by `whence`.
`ftell(fp)`	Returns offset position into file stream.
`stream_set_write_buffer (fp [, buffersize])`	Sets a buffer for file writing; the default is 8K.
`Is_dir(directory)`	Returns (and caches) `true` if named directory exists.
`Is_executable(file)`	Returns (and caches) `true` if named file is executable.
`Is_file(file)`	Returns (and caches) `true` if named file is a regular file.
`Is_link(file)`	Returns (and caches) `true` if named file is a symlink.
`Is_readable(file)`	Returns (and caches) `true` if named file is readable by PHP.
`is_writable (file/directory)`	Returns (and caches) `true` if named file or directory is writable by PHP.
`link(target, link)`	Creates hard link. Inoperative on Windows systems.
`linkinfo(path)`	Confirms existence of link. Inoperative on Windows systems.
`mkdir(path, mode)`	Makes directory at location *path* with the given permissions in octal mode.

Common Filesystem Function

Function	*Description*
`mkdir(path, mode)`	Makes directory at location *path* with the given permissions in octal mode.
`pclose(fp)`	Closes process file pointer opened by `popen()`.
`popen(command, mode)`	Opens process file pointer.
`readlink(link)`	Returns target of a symlink. Inoperative on Windows systems.
`rename(oldname, newname)`	Renames file.
`rewind(fp)`	Resets file pointer to beginning of file stream.
`rmdir(directory)`	Removes an empty directory.
`stat(file)`	Returns a selection of info about file.
`lstat(file)`	Returns a selection of info about file or symlink.
`symlink(target, link)`	Creates a symlink from target to link. Inoperative on Windows systems.
`touch(file, [time])`	Sets modification time; creates file if it does not exist.
`umask(mask)`	Returns `umask`, and sets to `mask & 0777`. With no argument passed, it simply returns the `umask`.

Network Function

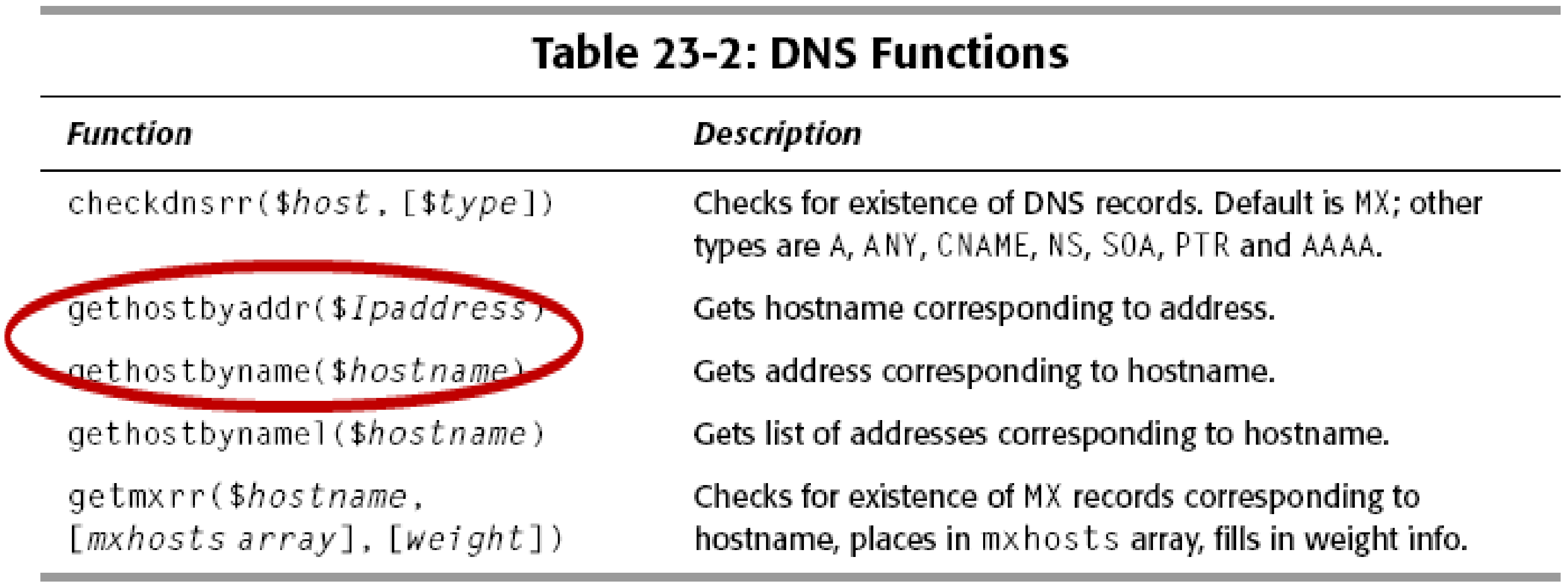


Table 23-2: DNS Functions

Function	*Description*
`checkdnsrr($host, [$type])`	Checks for existence of DNS records. Default is `MX`; other types are `A`, `ANY`, `CNAME`, `NS`, `SOA`, `PTR` and `AAAA`.
`gethostbyaddr($Ipaddress)`	Gets hostname corresponding to address.
`gethostbyname($hostname)`	Gets address corresponding to hostname.
`gethostbynamel($hostname)`	Gets list of addresses corresponding to hostname.
`getmxrr($hostname, [mxhosts array], [weight])`	Checks for existence of `MX` records corresponding to hostname, places in `mxhosts` array, fills in weight info.

Socket Function

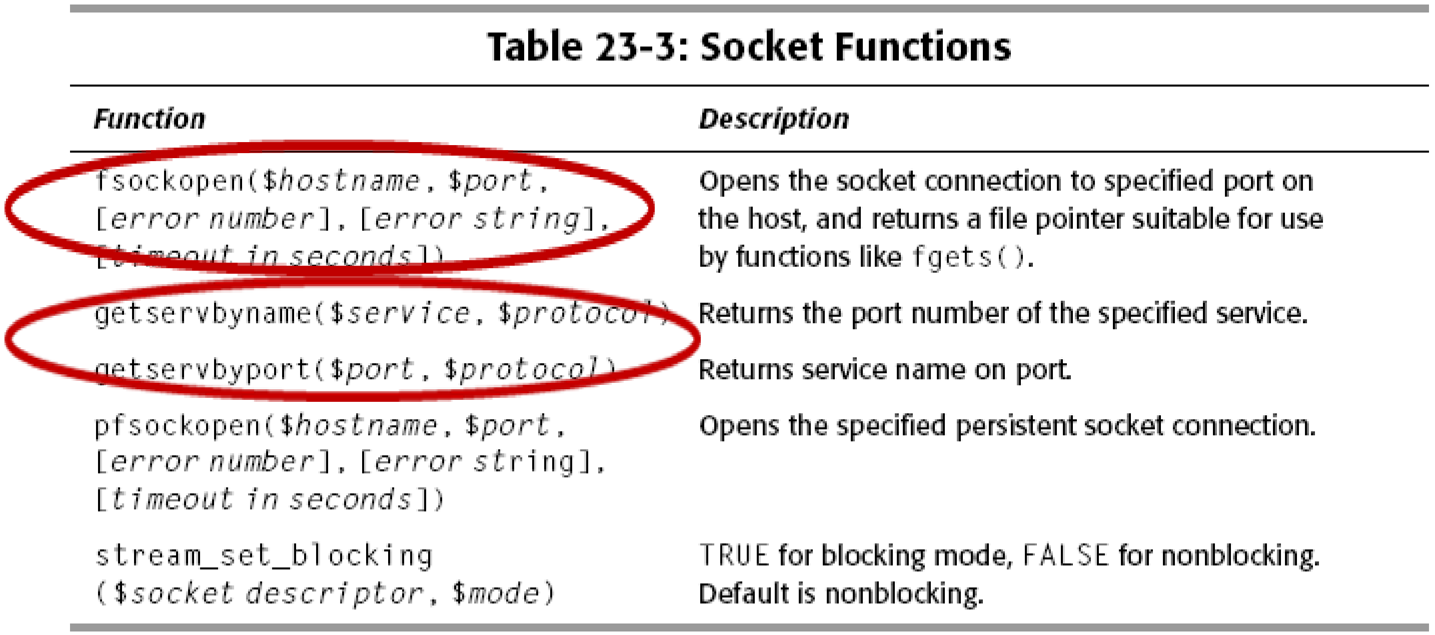


Table 23-3: Socket Functions

Function	*Description*
`fsockopen($hostname, $port, [error number], [error string], [timeout in seconds])`	Opens the socket connection to specified port on the host, and returns a file pointer suitable for use by functions like `fgets()`.
`getservbyname($service, $protocol)`	Returns the port number of the specified service.
`getservbyport($port, $protocol)`	Returns service name on port.
`pfsockopen($hostname, $port, [error number], [error string], [timeout in seconds])`	Opens the specified persistent socket connection.
`stream_set_blocking ($socket descriptor, $mode)`	`TRUE` for blocking mode, `FALSE` for nonblocking. Default is nonblocking.

File Upload : part11-3.htm

```
<form action="http://localhost/php2-5-file-upload.php" method="post“
        enctype="multipart/form-data“ >
……
   <input type="file" name="upload" value="browse" size="50"><br>
   <input type="submit" value=“send (上傳)"><br>
</form>
```

php2-5-file-upload.php

```
$upload_dir= "c:\\xampp\\htdocs\\tmp\\";
if ($_FILES['upload']['error'] == UPLOAD_ERR_OK) {
    if ( move_uploaded_file($_FILES['upload']['tmp_name'],
            $upload_dir.$_FILES['upload']['name']))  {
          echo "<BR>temp file name:".$_FILES['upload']['tmp_name'] ;
          echo "<BR>file name:".$_FILES['upload']['name'] ;
          echo "<BR>file type:".$_FILES['upload']['type'];
          echo "<BR>file size:".$_FILES['upload']['size'];
    }
}
```

File Management: Include & Require (1)

- Use the same set of functions across a set of Web site pages,
- Import the contents of some other files into the file being executed.
- Cloning function definitions at the beginning of each page
- For example
  - at the top of a PHP code file:

```
<?php include("filename.ext"); ?>
-----------------------------------------------
<?php
$LastName = "Park";
include("$LastName.inc");
?>
```

File Management: Include & Require (2)

- Difference: how they fail
  - If the file cannot be found, then
  - include construct will cause a warning to be printed, but processing of the script will continue;
  - require, on the other hand, will cause a fatal error if the file cannot be found.

Cookie (1)

- A cookie is a small piece of information that is retained on the client machine, either in the browser's application memory or as a small file written to the user's hard disk.
- It contains a name/value pair—setting a cookie means associating a value with a name and storing that pairing on the client side.
- Getting or reading a cookie means using the name to retrieve the value.

Cookie (2)

- A cookie is a special kind of file, located in the file system of your user’s browsing computer,
  - C:\Documents and Settings\WHLu\Cookies
- Web servers can read from and write to.
- Rather than checking for a passed GET/POST variable (and assigning a new identifier if none is found),
- your script checks the user’s machine for a previously written cookie file and stores a new identifier in a new cookie file if none is found or if the old cookie has expired.

Cookie (3): php2-6-cookie.php

- setcookie("name", "value", "expire", "path", "domain", "secure");
  - setcookie('membername', 'Tim');
  - setcookie('membername', 'Mary', time() + (60 * 60 * 24),"/", "www.troutworks.com", 1);
- Cookie access
  - $membername = $_COOKIE['membername'];
  - print("The member name is $membername<BR>");
- Delete cookie
  - setcookie('membername', 'xxx', time() -100);

Session (1)

- Keeping Track
  - tracking interactions with users over longer periods of time than it takes to generate a single Web page.
- Purpose
  - tracking how people navigate through a website
  - customize our users' experiences through a website
  - display advertisements to the user, but no more than once per session.
  - accumulate information about users' **actions** (e.g., an e-commerce site's shopping cart)
- Storing in **server side**: contain a name/value pair

Session (2)

- In the following session example, we perform the six tasks:

1. Initiate a session (or pick up an existing one): session_start().
2. Check for the existence of a pre-existing entry in $_SESSION. If not present, we assume that the session is new.
3. Increment a counter that tracks how many times that the user has visited this page.
4. Store the incremented counter back in $_SESSION.
    - $_SESSION['visit_count'] = visit_count+1;
5. Provide a link back to the page itself, embedding the session ID as an argument if it is found.
6. Clear the counter: unset($_SESSION['visit_count']);

Session (3): php2-3-session-1.php

```
<?php   session_start();    ?>
<HTML><HEAD><TITLE>Greetings</TITLE></HEAD>
<BODY> ……
<?php
if (  !IsSet($_SESSION['visit_count'])  ){
   print "Hello, you must have just arrived.  Welcome!<BR>";
   $_SESSION['visit_count'] = 1;
   echo "visit times: $_SESSION[visit_count]<br>";}
else {
   $visit_count = $_SESSION['visit_count'] + 1;
   $_SESSION['visit_count'] = $visit_count;
   echo "Back again are ya?  That makes $visit_count  times now <BR>";
}
//$self_url = $_SERVER['PHP_SELF'];
$self_url = "php2-3-session-2.php";
$session_id = session_id();
$href = "$self_url?s=$session_id";
$_SESSION['url'] = $href;
echo "SID: $session_id<br>";
echo "<BR><A HREF=\"$href\">Visit us again!</A>";
if   ($visit_count >=3)   {   unset( $_SESSION['visit_count'] );    }
?>
```