PHP

PHP Network Functions

Introduction [1]

- PHP Network functions gives to access internet into PHP
- There is no installation needed to use these functions; they are part of the PHP core.
- List of functions:
  - checkdnsrr
  - fsockopen
  - gethostbyaddr
  - Gethostbyname
  - ……. many more

gethostbyaddr() [1]

- It is used to get the internet host name which has given by IP Address
- Syntax: string gethostbyaddr (string $ip_address)
- Example 1
- Output:

  CSE49

  updateserver.nuni.ac.in

gethostbyname() [2]

- Get the IPv4 address corresponding to a given Internet host name
- Syntax: string gethostbyname (string $hostname)
- Example 1
  - Output:

    106.10.250.11

    127.0.0.1

gethostname() [1]

- It is used to get the host name
- It will return string value of host name of success or else gives False on failure
- Syntax: string gethostname (void)
- Example 1:
  - Output:

    The function returned a value: CSE49

Header() [2]

- It is used to send the row of HTTP Headers.
- Syntax: void header (string $string [, bool $replace = true [, int $http_response code]])

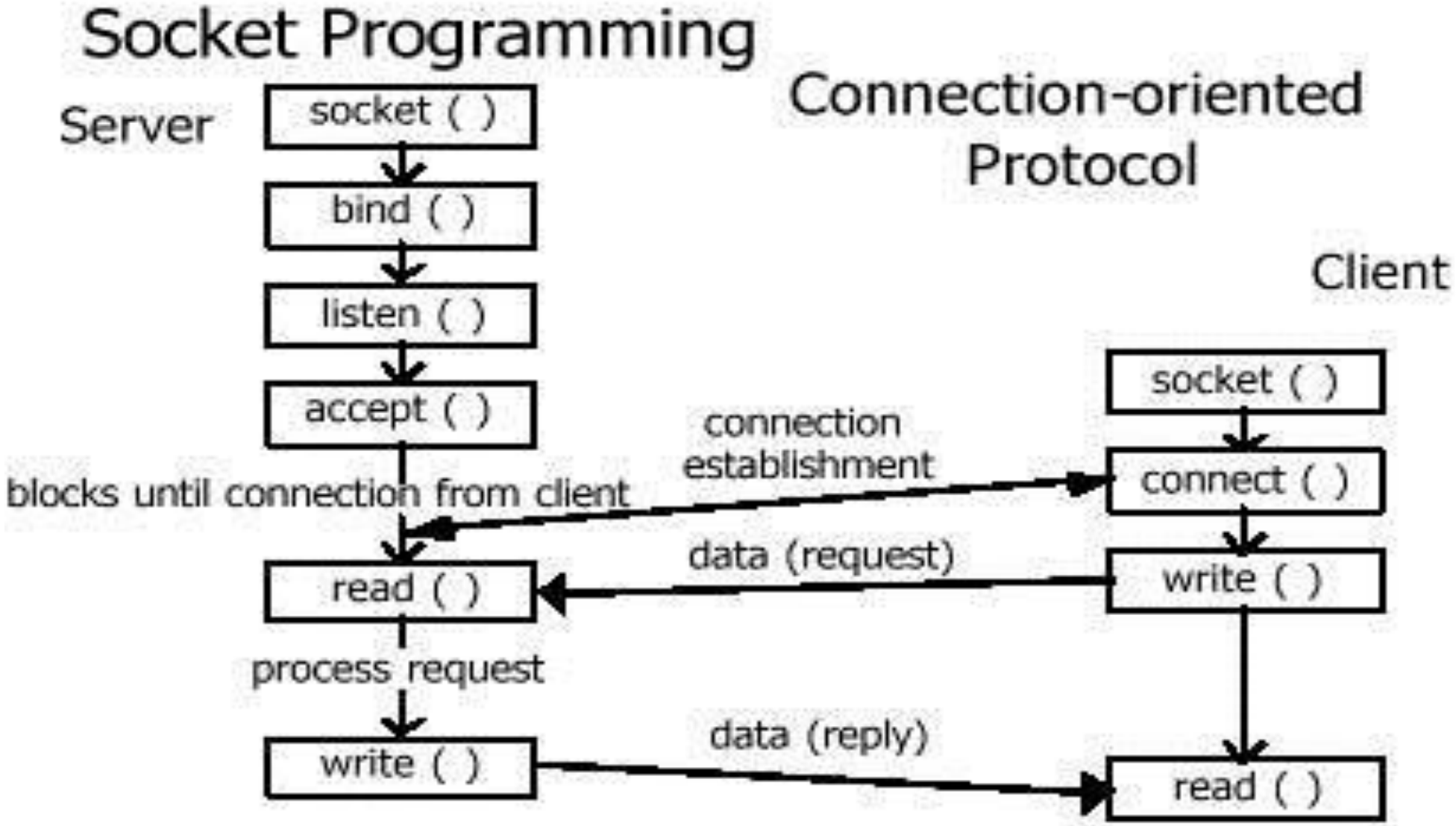
Sr.No	Parameters & Description
1	**string** It contains header string.
2	replace It replace parameter indicates whether the header should replace a previous similar header.
3	http_response_code Http response code is the specific values, it indicates whether response is success or not.

- Example 2
- Example 3

Socket Programming

- Enable socket programming in xampp:
  - Add following line in php.ini file

    extension=php_sockets.dll


Socket Programming
Connection-oriented Protocol
Server
socket ( )
bind ( )
listen ( )
accept ( )
blocks until connection from client
read ( )
process request
write ( )
Client
socket ( )
connect ( )
write ( )
read ( )
connection establishment
data (request)
data (reply)

- Source

set_time_limit() Function [2]

- set_time_limit — Limits the maximum execution time
- Set the number of seconds a script is allowed to run. If this is reached, the script returns a fatal error. The default limit is 30 seconds or, if it exists, the *max_execution_time* value defined in the php.ini.
- If set to zero, no time limit is imposed.

socket_create() Function [2]

- $sock = socket_create(AF_INET, SOCK_STREAM, 0);
- Function socket_create creates a socket and returns a socket descriptor which can be used in other network commands.
- The above code will create a socket with the following properties ...
  - Address Family : AF_INET (this is IP version 4)
  - Type : SOCK_STREAM (this means connection oriented TCP protocol)
  - Protocol : 0 [or IPPROTO_IP This is IP protocol]
- Apart from SOCK_STREAM type of sockets there is another type called SOCK_DGRAM which indicates the UDP protocol. This type of socket is non-connection socket.

socket_bind() Function [2]

- bool socket_bind (resource $socket , string $address [, int $port = 0])
- Parameters
  - socket
    - A valid socket resource created with socket_create().
  - address
    - If the socket is of the AF_INET family, the address is an IP in dotted-quad notation (e.g. 127.0.0.1).
    - If the socket is of the AF_UNIX family, the address is the path of a Unix-domain socket (e.g. /tmp/my.sock).
  - port (Optional)
    - The port parameter is only used when binding an AF_INET socket, and designates the port on which to listen for connections

socket_listen() [2]

- bool socket_listen (resource $socket [, int $backlog = 0])
- socket
  - A valid socket resource created with socket_create().
- backlog
  - A maximum of backlog incoming connections will be queued for processing. If a connection request arrives with the queue full the client may receive an error with an indication of ECONNREFUSED, or, if the underlying protocol supports retransmission, the request may be ignored so that retries may succeed.

socket_accept() [2]

- resource socket_accept (resource $socket)
- After the socket socket has been created using socket_create(), bound to a name with socket_bind(), and told to listen for connections with socket_listen(), this function will accept incoming connections on that socket. Once a successful connection is made, a new socket resource is returned, which may be used for communication

socket_write() [2]

- int socket_write (resource $socket , string $buffer [, int $length = 0])
- Parameters
  - socket
  - buffer
    - The buffer to be written.
  - length
    - The optional parameter length can specify an alternate length of bytes written to the socket. If this length is greater than the buffer length, it is silently truncated to the length of the buffer.
- Returns the number of bytes successfully written to the socket or FALSE on failure.

socket_read() [2]

- string socket_read (resource $socket , int $length [, int $type = PHP_BINARY_READ])
- Parameters
  - socket
    - A valid socket resource created with socket_create() or socket_accept().
  - length
    - The maximum number of bytes read is specified by the length parameter. Otherwise you can use \r, \n, or \0 to end reading (depending on the type parameter, see below).
  - type
    - Optional type parameter is a named constant:
      - PHP_BINARY_READ (Default) - use the system recv() function. Safe for reading binary data.
      - PHP_NORMAL_READ - reading stops at \n or \r.

socket_connect() [2]

- bool socket_connect (resource $socket , string $address [, int $port = 0])
- Initiate a connection to **address** using the socket resource **socket**, which must be a valid socket resource created with socket_create().
- Parameters
  - socket
  - address
    - The address parameter is either an IPv4 address in dotted-quad notation (e.g. 127.0.0.1) if socket is AF_INET, a valid IPv6 address (e.g. ::1) if IPv6 support is enabled and socket is AF_INET6 or the pathname of a Unix domain socket, if the socket family is AF_UNIX.
  - port
    - The port parameter is only used and is mandatory when connecting to an AF_INET or an AF_INET6 socket, and designates the port on the remote host to which a connection should be made.

socket_close() [2]

- void socket_close(resource $socket)
- socket_close() closes the socket resource given by socket. This function is specific to sockets and cannot be used on any other type of resources.

socket_last_error() [2]

- int socket_last_error ([resource $socket])
- If a socket resource is passed to this function, the last error which occurred on this particular socket is returned. If the socket resource is omitted, the error code of the last failed socket function is returned.

socket_strerror() [2]

- string socket_strerror (int $errno)
- socket_strerror() takes as its errno parameter a socket error code as returned by socket_last_error() and returns the corresponding explanatory text.

Error Handling [1]

- The default error handling in PHP is very simple. An error message with filename, line number and a message describing the error is sent to the browser.

Error Handling [1]

- different error handling methods:
  - Simple "die()" statements
  - Custom errors and error triggers
  - Error reporting

Error Handling [1]

- Using the die() function
  - Example

```
<?php
        if(!file_exists("welcome.txt")) {
                die("File not found");
        } else {
                $file=fopen("welcome.txt","r");
        }
?>
```

Error Handling [1]

- Custom Error Handler
  - Creating a custom error handler is quite simple. We simply create a special function that can be called when an error occurs in PHP.
  - This function must be able to handle a minimum of two parameters (error level and error message) but can accept up to five parameters (optionally: file, line-number, and the error context):

    error_function(error_level, error_message, error_file, error_line, error_context)

  - Example 6

Error Handling [1]

Syntax

```
error_function(error_level,error_message,
error_file,error_line,error_context)
```

Parameter	Description
error_level	Required. Specifies the error report level for the user-defined error. Must be a value number. See table below for possible error report levels
error_message	Required. Specifies the error message for the user-defined error
error_file	Optional. Specifies the filename in which the error occurred
error_line	Optional. Specifies the line number in which the error occurred
error_context	Optional. Specifies an array containing every variable, and their values, in use when the error occurred

Error Report levels

These error report levels are the different types of error the user-defined error handler can be used for:

Value	Constant	Description
2	E_WARNING	Non-fatal run-time errors. Execution of the script is not halted
8	E_NOTICE	Run-time notices. The script found something that might be an error, but could also happen when running a script normally
256	E_USER_ERROR	Fatal user-generated error. This is like an E_ERROR set by the programmer using the PHP function trigger_error()
512	E_USER_WARNING	Non-fatal user-generated warning. This is like an E_WARNING set by the programmer using the PHP function trigger_error()
1024	E_USER_NOTICE	User-generated notice. This is like an E_NOTICE set by the programmer using the PHP function trigger_error()
4096	E_RECOVERABLE_ERROR	Catchable fatal error. This is like an E_ERROR but can be caught by a user defined handle (see also set_error_handler())
8191	E_ALL	All errors and warnings (E_STRICT became a part of E_ALL in PHP 5.4)

Error Handling [1]

- Set Error Handler
  - The default error handler for PHP is the built in error handler.
  - We are going to make the function above the default error handler for the duration of the script.

```
set_error_handler("customError");
```

  - Since we want our custom function to handle all errors, the **set_error_handler()** only needed one parameter, a second parameter could be added to specify an error level.

Error Handling [1]

- Trigger an Error
  - In a script where users can input data it is useful to trigger errors when an illegal input occurs.
  - In PHP, this is done by the **trigger_error()** function.

  - An error can be triggered anywhere you wish in a script, and by adding a second parameter, you can specify what error level is triggered.

  - Possible error types:
    - E_USER_ERROR - Fatal user-generated run-time error. Errors that can not be recovered from. Execution of the script is halted
    - E_USER_WARNING - Non-fatal user-generated run-time warning. Execution of the script is not halted
    - E_USER_NOTICE - Default. User-generated run-time notice. The script found something that might be an error, but could also happen when running a script normally
  - Example 7, Example 8

Exception Handling [5]

- Exceptions are used to change the normal flow of a script if a specified error occurs.

Exception Handling [1]

- This is what normally happens when an exception is triggered:
  - The current code state is saved
  - The code execution will switch to a predefined (custom) exception handler function
  - Depending on the situation, the handler may then resume the execution from the saved code state, terminate the script execution or continue the script from a different location in the code
- We will show different error handling methods:
  - Basic use of Exceptions
  - Creating a custom exception handler
  - Multiple exceptions
  - Re-throwing an exception
  - Setting a top level exception handler

Exception Handling [1]

- **Try, throw and catch**
  - Example 9

- **Creating a Custom Exception Class**
  - To create a custom exception handler you must create a special class with functions that can be called when an exception occurs in PHP. The class must be an extension of the exception class.
  - The custom exception class inherits the properties from PHP's exception class and you can add custom functions to it.
  - Example 10

- **Multiple Exceptions**
  - Example 11

Exception Handling [1]

- **Re-throwing Exceptions**
  - Example 13

- **Set a Top Level Exception Handler**
  - The **set_exception_handler()** function sets a user-defined function to handle all uncaught exceptions
  - Example 12

Exception Handling [1]

- Exceptions are used to change the normal flow of a script if a specified error occurs.
-

References

1. https://www.tutorialspoint.com/php/php_network_functions.htm
2. http://php.net
3. https://www.codeproject.com/Tips/418814/Socket-Programming-in-PHP
4. https://www.binarytides.com/php-socket-programming-tutorial/
5. https://www.w3schools.com/php/php_exception.asp

Thank you….