Chapter

13

외부 처리

PHP는 시스템의 외부 명령어를 실행하고, 결과값을 반환받을 수 있는 몇 개의 시스템 함수들을 제공합니다.

외부 명령 처리는 시스템의 커맨드 명령 창에서 실행하는 것과 동일한 처리를 의미합니다.

13.1 시스템 함수

PHP는 시스템의 명령을 호출하여 실행할 수 있습니다. PHP는 세 가지 방식으로 시스템을 호출하여 실행할 수 있습니다.

| 내장 함수 |

```
string exec ( string $command [, array &$output [, int &$return_var ]] )
```

내장 함수 exec()는 시스템의 외부 **명령을 실행**합니다. exec()는 반환값으로는 처리 결과 문자열이 반환됩니다.

output 인수가 있으면 지정된 배열은 명령의 모든 출력 행을 배열로 채워집니다. 하지만 \ n 같은 후행 공백은 이 배열에 포함되지 않습니다. 만일 output 변수가 초기화되지 않은 값인 경우에 배열 뒤에 추가로 연결됩니다.

예제 파일 | **exec.php**

```
<?php
   $result = exec("dir",$output);
   echo $result;
   echo "<br>";
   print_r($output);

?>
```

화면 출력

```
2개 디렉터리 84,432,617,472바이트 남음
Array ( [0] => C 드라이브의 볼륨: windows [1] => 볼륨 일련번호: 447E-0DB0 [2] => [3]
=> C:\php-7.1.4-Win32-VC14-x86\sample1 디렉터리 [4] => [5] => 2017-08-17 오후
06:06
. [6] => 2017-08-17 오후 06:06
.. [7] => 2017-08-17 오후 06:04 95 exec.php [8] => 1개 파일 95바이트 [9] => 2개 디렉
터리 84,432,617,472바이트 남음
```

| 내장 함수 |

```
string system ( string $command [, int &$return_var ] )
```

내장 함수 system()은 외부 프로그램을 실행합니다. 실행 후 **결과를 출력**합니다. system()은 주어진 명령을 실행하고 결과를 출력한다는 점에서 함수의 C 버전과 같습니다.

예제 파일 | **system.php**

```
<?php
   $result = system("dir",$output);
   echo $result;

?>
```

화면 출력

```
C 드라이브의 볼륨: windows 볼륨 일련번호: 447E-0DB0 C:\php-7.1.4-Win32-VC14-x86\
sample1 디렉터리 2017-08-17 오후 06:20
. 2017-08-17 오후 06:20
.. 2017-08-17 오후 06:04 95 exec.php 2017-08-17 오후 06:20 66 system.php 2개 파일
161바이트 2개 디렉터리 84,477,603,840바이트 남음 2개 디렉터리 84,477,603,840바이트 남음
```

| 내장 함수 |

string **shell_exec** (string $cmd)

내장 함수 shell_exec()는 셸을 통해 명령을 실행합니다. 실행한 전체 출력을 문자열로 반환합니다.

예제 파일 | **shell_exec.php**

```
1  <?php
2     $output = shell_exec('ls -lart');
3     echo "<pre>$output</pre>";
4
5  ?>
```

13.2 프로세스

| 내장 함수 |

resource **proc_open** (string $cmd , array $descriptorspec , array &$pipes [, string $cwd [, array $env [, array $other_options]]])

내장 함수 proc_open()은 명령 실행과 입/출력에 대한 파일 포인터를 생성합니다. proc_open()은 popen()과 비슷한 점이 많지만 좀 더 유연한 제어를 할 수 있습니다.

cmd는 실행할 명령어입니다.

두 번째 인자 descriptorspec는 인덱스 타입의 배열입니다. 인덱스 키는 자식 프로세스에게 전달되는 방법을 설정합니다. 0은 stdin, 1은 stdout, 2는 stderr입니다.

pipes는 생성된 파이프의 PHP 끝 부분에 해당하는 파일 포인터의 인덱스 배열로 설정됩니다.

cwd는 명령을 실행하는 초기 작업 디렉터리입니다. 이 값은 상대 경로가 아닌 절대 경로 형태로 지정해야 합니다. 또는 기본 PHP가 설치된 작업 디렉터리를 사용할 경우에는 NULL을 사용하면 됩니다.

env는 실행되는 명령어에 대한 환경 변수를 배열로 설정합니다.

other_options

추가 옵션을 지정할 수 있습니다. 현재 지원되는 옵션은 다음과 같습니다.

- suppress_errors(윈도우 전용): 이 함수가 TRUE로 설정된 경우 생성된 오류를 억제합니다.
- bypass_shell(윈도우 전용): TRUE로 설정하면 cmd.exe 셸을 무시합니다.

| 내장 함수 |

```
int proc_close ( resource $process )
```

내장 함수 proc_open()에 의해 열린 프로세스를 닫습니다. 프로세스의 종료 코드를 반환합니다.

- command: proc_open()에 전달된 명령 문자열입니다.
- pid: process id
- running: 프로세스가 아직 실행 중이면 TRUE, 종료된 경우 FALSE 값을 반환합니다.
- signaled: 자식 프로세스가 포착되지 않은 신호에 의해 종료된 경우 TRUE 값입니다. 윈도우에서는 항상 FALSE 값으로 설정됩니다.

- stopped: 자식 프로세스가 신호에 의해 중단 된 경우 TRUE 값 입니다. 윈도우에서는 항상 FALSE로 설정됩니다.
- exitcode: 실행 도중 프로세스가 반환한 종료 코드로 FALSE 값입니다. 첫 번째 호출만 실제 값을 반환하고, 다음 호출은 −1을 반환합니다.
- termsig: 자식 프로세스가 실행을 종료하게 만든 신호의 번호입니다
- stopsig: 중지된 경우 자식 프로세스가 실행을 멈추게 만든 신호의 번호입니다.

| 내장 함수 |

```
bool proc_nice ( int $increment )
```

내장 함수 proc_nice()는 현재 프로세스의 우선순위 변경합니다.

proc_nice()는 increment에 지정된 양만큼 현재 작업 중인 프로세스의 우선순위를 변경하게 됩니다. 양수 값의 증가는 현재 프로세스의 우선순위를 낮추고, 음수 값의 증가는 우선순위를 높입니다.

| 내장 함수 |

```
bool proc_terminate ( resource $process [, int $signal = 15 ] )
```

내장 함수 proc_terminate()는 proc_open에 의해 오픈된 프로세스를 종료합니다. proc_open() 함수를 통해서 생성된 프로세스를 종료합니다. proc_terminate() 호출 시 프로세서는 기다리지 않고 즉시 종료합니다.

예제 파일 | **proc.php**

```
1  <?php
2      $descriptorspec = array(
3          0 => array("pipe", "r"),  // stdin
4          1 => array("pipe", "w"),  // stdout
5          2 => array("file", "/tmp/error-output.txt", "a") // stderr
6      );
```

```
7
8       $cwd = '/tmp';
9       $env = array('some_option' => 'aeiou');
10
11      $process = proc_open('php', $descriptorspec, $pipes, $cwd, $env);
12      echo "proc status<br>";
13      print_r(proc_get_status($process));
14      echo "<br>";
15
16      if (is_resource($process)) {
17          // $pipes now looks like this:
18          // Any error output will be appended to /tmp/error-output.txt
19
20          echo "0 => writeable handle connected to child stdin <br>";
21          $php_code = "<?php echo 'hello' ?>";
22          fwrite($pipes[0], $php_code);
23          fclose($pipes[0]);
24
25          echo "1 => readable handle connected to child stdout <br>";
26          echo stream_get_contents($pipes[1]);
27          echo "<br>";
28          fclose($pipes[1]);
29
30          // It is important that you close any pipes before calling
31          // proc_close in order to avoid a deadlock
32          $return_value = proc_close($process);
33
34          echo "command returned $return_value <br>";
35      }
36
37  ?>
```

화면 출력

```
proc status
Array ( [command] => php [pid] => 28647 [running] => 1 [signaled] => [stopped]
=> [exitcode] => -1 [termsig] => 0 [stopsig] => 0 )
0 => writeable handle connected to child stdin
1 => readable handle connected to child stdout
hello
command returned 0
```