jinyPHP 3권 - 16. 기타함수

**16. 함수**

====================

PHP는 수많은 함수들을 포함하고 있습니다. 또한, 다양한 함수들을 통하여 PHP 응용프로그램을 개발합니다.

함수들은 내부적으로 제공함수와 사용자가 직접 작성하여 생성한 함수로 구분할 수 있습니다. 또한 함수의 명은 중복해서 사용을 할 수 없습니다. PHP는 내부적으로 함수를 관리하고 처리할 수 있는 별도의 특수 함수들을제공합니다.

**16.1 함수 목록**

====================

함수를 작성하거나 사용할 때 함수의 이름은 매우 중용합니다. 함수는 PHP 내에서 중복해서 사용을 할 수 없습니다. 만일 중복된 함수명을 생성하면 오류를 발생합니다.

또한 존재하지 않은 함수를 호출할때도 PHP는 오류를 발생합니다. PHP는 지정한 이름의 함수가 존재하는지 확인할 수 있는 특별한 함수를 제공합니다.

**|내부함수|**

|  |
| --- |
| bool **function\_exists** ( string $function\_name ) |

내부함수 function\_exists()는 스크립트 내에 주어진 이름의 함수의 정의 되어 있는지를 확인합니다.

**예제) function\_exists.php**

|  |
| --- |
| <?php  if (function\_exists('str\_replace')) {  echo "함수가 정의 되어 있습니다.<br />\n";  } else {  echo "없는 함수 있습니다.<br />\n";  }  ?> |

**화면출력)**

함수가 정의 되어 있습니다.

**|내부함수|**

|  |
| --- |
| array **get\_defined\_functions** ([ bool $exclude\_disabled = FALSE ] ) |

내부함수 get\_defined\_functions()는 정의 된 모든 함수의 목록을 배열로 반환합니다.

**예제) get\_defined\_functions.php**

|  |
| --- |
| <?php  $arr = get\_defined\_functions();  print\_r($arr);  ?> |

**16.2 함수 인자**

====================

우리가 함수를 이용하는 것은 코드의 중복되는 처리들을 블록화 하는 것입니다. 함수는 처리를 위한 기본 값의 매개변수와 반환 값들을 가지고 있습니다.

PHP는 유연한 함수의 매개변수 처리를 지원합니다. 이와 관련하여 매개변수의 처리를 할 수 있는 몇 개의 함수들을 제공합니다.

**|내부함수|**

|  |
| --- |
| int **func\_num\_args** ( void ) |

내부함수 func\_num\_args()는 함수에 전달 된 인수의 갯 수를 반환합니다.

**예제) func\_num\_args.php**

|  |
| --- |
| <?php  function foo()  {  $numargs = func\_num\_args();  echo "Number of arguments: $numargs\n";  }  foo(1, 2, 3);  ?> |

**화면출력)**

Number of arguments: 3

위의 실험에서 사용자 지정함 함수의 매개변수를 지정하지 않았습니다. 그리고 3개의 값을 전달하여 함수를 호출하였습니다. func\_num\_args()를 통하여 전달되는 인자의 개수를 확인할 수 있습니다.

**|내부함수|**

|  |
| --- |
| mixed **func\_get\_arg** ( int $arg\_num ) |

내부함수 func\_get\_arg()는 인수 목록에서 항목 반환 합니다.

**예제) func\_get\_arg.php**

|  |
| --- |
| <?php  function foo()  {  $numargs = func\_num\_args();  echo "Number of arguments: $numargs <br>";  if ($numargs >= 2) {  echo "Second argument is: " . func\_get\_arg(1) . " <br>";  }  }  foo(1, 2, 3);  ?> |

**화면출력)**

Number of arguments: 3

Second argument is: 2

**|내부함수|**

|  |
| --- |
| array **func\_get\_args** ( void ) |

내부함수 func\_get\_args()는 함수의 인수 목록을 구성하는 배열을 반환합니다.

**예제) func\_get\_args.php**

|  |
| --- |
| <?php  function foo()  {  $numargs = func\_num\_args();  echo "Number of arguments: $numargs <br>";  if ($numargs >= 2) {  echo "Second argument is: " . func\_get\_arg(1) . "<br>";  }    $arg\_list = func\_get\_args();  for ($i = 0; $i < $numargs; $i++) {  echo "Argument $i is: " . $arg\_list[$i] . "<br>";  }  }  foo(1, 2, 3);  ?> |

**화면출력)**

Number of arguments: 3

Second argument is: 2

Argument 0 is: 1

Argument 1 is: 2

Argument 2 is: 3

**16.3 콜백호출**

====================

**|내부함수|**

|  |
| --- |
| mixed **call\_user\_func\_array** ( callable $callback , array $param\_arr ) |

내부함수 call\_user\_func\_array()는 매개 변수 배열을 사용하여 콜백 호출합니다.

**예제) call\_user\_func\_array.php**

|  |
| --- |
| <?php  // 함수 호출  function foobar($arg, $arg2) {  echo \_\_FUNCTION\_\_, " got $arg and $arg2 <br>";  }  // 함수명  // 매개변수 매열  call\_user\_func\_array("foobar", array("one", "two"));  // 객체 매소드 호출  class foo {  function bar($arg, $arg2) {  echo \_\_METHOD\_\_, " got $arg and $arg2<br>";  }  }  $foo = new foo;  // 인스턴스, 매소드 배열  // 매개변수 매열  call\_user\_func\_array(array($foo, "bar"), array("three", "four"));  ?> |

**화면출력)**

foobar got one and two

foo::bar got three and four

**|내부함수|**

|  |
| --- |
| mixed **call\_user\_func** ( callable $callback [, mixed $parameter [, mixed $... ]] ) |

내부함수 call\_user\_func()는 매개 변수에 의해 콜백 호출 처리합니다.

**예제) call\_user\_func.php**

|  |
| --- |
| <?php  function barber($type)  {  echo "You wanted a $type haircut <br>";  }  call\_user\_func('barber', "mushroom");  call\_user\_func('barber', "shave");  ?> |

**화면출력)**

You wanted a mushroom haircut

You wanted a shave haircut

**16.4 매서드 호출**

====================

**|내부함수|**

|  |
| --- |
| mixed **forward\_static\_call** ( callable $function [, mixed $parameter [, mixed $... ]] ) |

내부함수 forward\_static\_call()는 정적 메서드 호출 합니다.

**예제) forward\_static\_call .php**

|  |
| --- |
| <?php  class A  {  const NAME = 'A';  public static function test() {  $args = func\_get\_args();  echo static::NAME, " ".join(',', $args)." <br>";  }  }  class B extends A  {  const NAME = 'B';  public static function test() {  echo self::NAME, "<br>";  forward\_static\_call(array('A', 'test'), 'more', 'args');  forward\_static\_call( 'test', 'other', 'args');  }  }  B::test('foo');  function test() {  $args = func\_get\_args();  echo "C ".join(',', $args)." <br>";  }  ?> |

**화면출력)**

B

B more,args

C other,args

**|내부함수|**

|  |
| --- |
| mixed **forward\_static\_call\_array** ( callable $function , array $parameters ) |

내부함수 forward\_static\_call\_array()는 정적 메서드를 호출합니다. 인수는 배열로 전달합니다.

**예제) forward\_static\_call\_array.php**

|  |
| --- |
| <?php  class A  {  const NAME = 'A';  public static function test() {  $args = func\_get\_args();  echo "Class == ".static::NAME, " ".join(',', $args)." <br>";  }  }  class B extends A  {  const NAME = 'B';  public static function test() {  echo "Class : ". self::NAME, "<br>";  // A클래스 test 매소드 호출  forward\_static\_call\_array(array('A', 'test'), array('more', 'args'));  // 함수호출  forward\_static\_call\_array( 'test', array('other', 'args'));  }  }  B::test('foo');  function test() {  $args = func\_get\_args();  echo "function call = ".join(',', $args)." <br>";  }  ?> |

**화면출력)**

Class : B

Class == B more,args

function call = other,args

**16.5 틱 실행**

====================

**|내부함수|**

|  |
| --- |
| bool **register\_tick\_function** ( callable $function [, mixed $arg [, mixed $... ]] ) |

내부함수 register\_tick\_function()는 틱에서 실행될 함수를 등록합니다.

**예제) register\_tick\_function.php**

|  |
| --- |
| <?php  declare(ticks=1);  register\_tick\_function('my\_function', true);  $object = new my\_class();  register\_tick\_function(array(&$object, 'my\_method'), true);  ?> |

**|내부함수|**

|  |
| --- |
| void **unregister\_tick\_function** ( string $function\_name ) |

내부함수 unregister\_tick\_function()는 틱에서 실행될 함수를 해제합니다.