jinyPHP 3권 - 3. 문자열

**03. 문자열**

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

문자열은 응용프로그램을 개발하면서 가장 많이 사용되는 기능입니다. 프로그램의 복잡한 동작과 수치적인 연산을 하지만, 최종적으로 사용자에게 보여지는 모든 것은 글자화된 문자열 입니다.

PHP는 문자들을 다루고 처리할 수 있도록 다양한 문자열 함수들을 제공하고 있습니다. 일부 내부함수들은 추가 모듈설치가 필요할 수도 있습니다.

**03.1 문자열변수**

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

PHP는 C언어 스타일의 인터프리터 언어 입니다. 따라서 기본적인 문자열의 처리와 개념 또한 C언어의 속성과 스타일을 따르는 경우가 많습니다.

PHP에서 문자열을 조작하고 처리하는 것은 생각보다 매우 쉽습니다. C언어처럼 변수의 타입과 엄격한 문자와 문자열을 처리할 수 있는 규칙들을 적용하지는 않습니다. 또한, PHP는 변수에 문자열을 입력함으로써 자동적으로 문자열 변수형의 타입으로 설정됩니다.

**03.1.1 문자열 길이**

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

문자열의 데이터를 처리할 때 가장 많이 사용을 하는 기능은 문자열의 길이를 측정하는 것입니다. 문자열은 여러 개의 문자들의 집합이기 때문에 총 몇 개의 글자로 구성이 되어 있는지 확인하는 것이 중요합니다.

문자열의 길이를 기반으로 문자열을 처리해야 하는 패턴의 처리량이 결정됩니다. C언어와 같은 언어에서는 문자열을 처리할 때 배열을 통하여 저장을 많이 합니다. 실제적으로 PHP 내부에서도 변수에 문자열을 저장하면 배열 형태로 처리할 수 있습니다.

**$msg = “hello world!”;**

PHP에서 위와 같이 변수의 생성과 문자열을 입력합니다. 변수 $msg 는 문자열의 데이터를 가지고 있기 때문에 문자열 변수라고 할 수 있습니다. 그리고 $msg 변수는 배열의 접근 방식으로 한글자 한글자를 지정할 수 있습니다.

배열과 같은 특성을 가진 문자열 변수는 길이를 측정할 수 있습니다. 한글자씩 문자를 포함하고 있는 메모리의 크기를 계산하면 되기 때문입니다. PHP에서는 문자열의 길이를 측정할 수 있는 내부함수를 제공하고 있습니다.

**|내부함수|**

|  |
| --- |
| int **strlen** ( string $string ) |

내부함수 strlen()는 변수에 들어있는 문자열의 길이를 확인할 수 있습니다. 함수의 매개변수 값으로 길이를 측정할 변수명을 입력하면 됩니다. 문자열의 길이는 정수형태의 값으로 반환을 합니다. 문자열은 문자들의 연결된 배열의 크기를 측정하는 것과 같습니다.

**예제파일) strlen-01.php**

|  |
| --- |
| <?php  // 문자열 변수를 생성합니다.  $name = "Hello world!";  // 문자의 갯수를 출력합니다.  echo "문자의 갯수는 :".**strlen($name)**;  ?> |

**화면출력)**

문자의 갯수는 :12

문자열의 변수가 배열형태를 띠고 있고, strlen() 함수를 통하여 문자열의 길이를 측정할 수 있었습니다. 이러한 특성을 이용하여 문자열 변수에서 한글자씩 문자값을 불러서 개별적으로 처리도 가능합니다.

**예제파일) strlen-02.php**

|  |
| --- |
| <?php  // 문자열 변수를 생성합니다.  $name = "Hello world!";  // 한줄에 문자 한자씩 출력을 합니다.  for ($i=0;$i<strlen($name);$i++) {  echo **$name[$i]**."<br>";  }  ?> |

화면출력)

H

e

l

l

o

w

o

r

l

d

!

위의 실험은 문자열의 변수를 배열로 접근하여 한글자씩 출력을 합니다. 문자열을 배열형태로 접근을 할때는 문자열 변수명 뒤에 대괄호 [] 를 통하여 접근을 합니다. 대괄호 안에 접근값으로 0부터 시작되는 인덱스 값을 넣어주시면 됩니다.

**03.1.2 문자열 카운트**

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

문자열안에 또다른 문자열을 포함하고 있을 경우, 포함되고 있는 문자의 반복 갯수를 측정할 수 있습니다.

**|내부함수|**

|  |
| --- |
| int **substr\_count** ( string $haystack , string $needle [, int $offset = 0 [, int $length ]] ) |

내부함수 substr\_count()는 원본의 $haystack 변수에서 검색하고자 하는 문자열 $needle 값이 몇 개를 포함하고 있는지를 확인합니다. 반환값은 정수입니다. 검색을 할때는 대소문자를 구분하여 처리를 합니다.

**예제파일) substr\_count.php**

|  |
| --- |
| <?php  $text = 'This is a test';  echo $text . "<br>";  echo "원본 문자열 길이 = ". strlen($text) ."<br>";  // 출력: 14  // 'This is a test' 안에 is 두번 존재  echo "needle substring이 발생하는 횟수 =". **substr\_count($text, 'is')**."<br>";  // 출력 :2    // 오프셋 3 적용하여 문자열은 's is a test' 형태로 앞에 3개의 문자열은 건너 뜁니다.  echo "오프셋 적용 = ".substr\_count($text, 'is', 3) ."<br>";  // 오프셋 3 적용 및 문자열의 길이는 3으로 제한합니다.  // 따라서 문자열은 's i' 로 적용됩니다.  echo "오프셋 적용, 길이제한 = ".substr\_count($text, 'is', 3, 3) ."<br>";  // 오류발생  // 오프셋 5 + 길이 10 = 총길이 15로 원본 $text 문자열 14보다 크기 때문에 오류 발생  echo substr\_count($text, 'is', 5, 10) ."<br>";  // prints only 1, because it doesn't count overlapped substrings  $text2 = 'gcdgcdgcd';  echo substr\_count($text2, 'gcdgcd') ."<br>";  ?> |

**화면출력)**

This is a test

원본 문자열 길이 = 14

needle substring이 발생하는 횟수 =2

오프셋 적용 = 1

오프셋 적용, 길이제한 = 0

1

위의 실험은 $text 문자열에서 “is” 의 문자열을 포함하고 있는 문자열의 개수를 찾아서 처리를 합니다.

**03.1.3 단어갯수 측정**

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

문자열은 하나의 단어일수도 있고, 문장일 수도 있습니다. 만일 문자열이 여러 개의 단어로 구성되어 있는 문장일 경우, 포함하고 있는 단어의 개수를 측정할 수 있습니다.

**|내부함수|**

|  |
| --- |
| mixed **str\_word\_count** ( string $string [, int $format = 0 [, string $charlist ]] ) |

내부함수 str\_word\_count()은 입력된 문자열에서 단어의 개수를 찾아서 개수를 반환합니다.

**예제파일) str\_word\_count.php**

|  |
| --- |
| <?php  $str = "I love you";  echo **str\_word\_count($str)**;  ?> |

**화면출력)**

3

**03.2 문자열 자르기**

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

문자열을 처리하는데 있어서 많이 사용하는 기능은 문자열을 분리하는 것입니다. 문자열은 다양한 형태로 존재하기 때문에 관련함수 들이 많이 지원 됩니다.

**03.2.1 문자열 제거**

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

내부함수 substr()는 입력한 문자열의 일부분을 잘라낼수 있습니다. 문자열을 잘라 낼때는 문자열의 시작위치와 끝 위치를 지정하여 부분을 추출합니다.

**|내부함수|**

|  |
| --- |
| string **substr** ( string $string , int $start [, int $length ] ) |

**예제파일) substr.php**

|  |
| --- |
| <?php  $string = "abcdefghijklmnopqrstuvwxyz";  // 1위치 이후부터 표시함  echo **substr($string,1)**."<br>";  // 3위치 이후부터 5개 표시함  echo **substr($string,3,5)**."<br>";  ?> |

**화면출력)**

bcdefghijklmnopqrstuvwxyz

defgh

위의 실험은 문자열 $string 변수에서 특정 부분의 문자열을 추출합니다. 문자열을 추출할 때 세번째 끝값은 생략이 가능합니다. 끝값을 생략할때는 시작부터 끝까지를 모두 출력합니다.

**03.2.2 공백제거**

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

문자열 변수는 공백도 문자로 취급을 합니다. 문자열을 가공하다가 보면 문자열 앞에 또는 뒤에 쓸모가 없는 공백문자들이 있는 경우가 있습니다. 이러한 경우 공백을 자동적으로 제거할 수 있는 내부함수를 제공하고 있습니다.

PHP는 공백을 제거하고 처리할 수 있는 3개의 함수를 제공합니다. trim(), ltrim(), rtrim() 함수는 문자열의 공백을 제거합니다.

**|내부함수| 양쪽공백 제거**

|  |
| --- |
| string **trim** ( string $str [, string $character\_mask = " \t\n\r\0\x0B" ] ) |

내부함수 trim()는 문자열의 앞/뒤에 존재하는 공백문자를 제거합니다.

**|내부함수| 우측공백 제거**

|  |
| --- |
| string **rtrim** ( string $str [, string $character\_mask ] )  string **chop** (string str); |

내부함수 rtrim(), chop()는 문자열 뒤쪽에 있는 공백문자를 제거합니다. chop() 함수는 rtrim() 함수의 별칭입니다.

**|내부함수| 좌측공백 제거**

|  |
| --- |
| string **ltrim** ( string $str [, string $character\_mask ] ) |

내부함수 ltrim()는 문자열의 오른쪽 공백문자를 제거합니다.

**예제파일) trim.php**

|  |
| --- |
| <?php  $string = " jiny ";  echo "안녕하세요!".**$string**."입니다.<br>";  // 앞뒤 공백 문자열을 삭제합니다.  echo "안녕하세요!".**trim($string)**."입니다.<br>";  // 오른쪽 공백을 제거합니다.  echo "안녕하세요!".**chop($string)**."입니다.<br>";  echo "안녕하세요!".**rtrim($string)**."입니다.<br>";  // 왼쪽 공백을 제거합니다.  echo "안녕하세요!".**ltrim($string)**."입니다.<br>";  ?> |

**화면출력)**

안녕하세요! jiny 입니다.

안녕하세요!jiny입니다.

안녕하세요! jiny입니다.

안녕하세요! jiny입니다.

안녕하세요!jiny 입니다.

**03.3 문자열 검색**

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

문자열을 다양하게 처리를 하기 위해서는 문자열에서 특정 문자들의 위치를 찾아 처리하는 것입니다. 문자열의 내용을 검색하고 값을 반환하거나 위치값을 구할 수 있습니다.

**03.3.1 첫번째 검색**

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

내부함수 strstr()는 문자열안에서 찾을 문자의 첫 번째 위치를 구하고, 이후의 문자열을 데이터를 반환합니다. 문자를 검색할때는 대소문자를 구별을 하는 점을 주의 하셔야 합니다.

**|내부함수|**

|  |
| --- |
| string **strstr** ( string $haystack , mixed $needle [, bool $before\_needle = false ] ) |

strchr() 함수는 strstr() 함수의 별칭으로 동일한 작업을 수행합니다.

**예제파일) strstr.php**

|  |
| --- |
| <?php  $str = "abcdeghijklem-abcdeghijklem-1234567";  echo "원본 : " . $str . "<br>";  // 문자열에서 em- 으로 시작하는 위치를 찾아  // 이후의 문자열을 반환합니다.  $a = **strstr($str,"em-")**;  echo $a;  ?> |

**화면출력)**

원본 : abcdeghijklem-abcdeghijklem-1234567

em-abcdeghijklem-1234567

**|내부함수|**

|  |
| --- |
| string **stristr** ( string $haystack , mixed $needle [, bool $before\_needle = false ] ) |

내부함수 stristr()은 strstr() 함수와 동일한 동작을 하는 함수입니다. 하지만 차이점으로는 대소문자를 구별하지 않습니다.

**03.3.2 마지막 검색**

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

내부함수 strrchr()는 strstr() 함수와 반대로 마지막 위치를 찾아서 문자열 데이터를 처리합니다. 검색하고자 하는 문자열에 찾을 문자들이 여러 개가 있는 경우 마지막 부분을 반환합니다.

**|내부함수|**

|  |
| --- |
| string **strrchr** ( string $haystack , mixed $needle ) |

**예제파일) strrchr.php**

|  |
| --- |
| <?php  $str = "abcdeghijkl**em**-abcdeghijkl**em-1234567**";  echo strrchr($str,"em");  ?> |

**화면출력)**

em-1234567

위 실험에서 원본의 문자열에서 em 글자를 찾습니다. em 글자는 원본 문자열에 2번 포함하고 있습니다. 마지막의 em 위치를 찾아서 이후의 문자열의 데이터를 반환을 합니다.

**03.3.3 첫번째 위치**

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

내부함수 strpos()는 문자열에서 검색하고자 하는 글자를 찾아 첫번째 위치를 반환합니다. 만일 글자를 찾게 되면 문자열의 시작 위치를 정수값으로 반환합니다.

**|내부함수|**

|  |
| --- |
| mixed **strpos** ( string $haystack , mixed $needle [, int $offset = 0 ] ) |

**예제파일) strpos.php**

|  |
| --- |
| <?php  $string = "abcdefg";  $keyword = "cde";  if (($pos = strpos($string, $keyword)) === false) {  echo "Err] 찾는 문자열이 없습니다.";    } else {  echo "문자열 시작위치 $pos 존재<br>";  }  ?> |

**화면출력)**

문자열 시작위치 2 존재

**|내부함수|**

|  |
| --- |
| mixed **stripos** ( string $haystack , string $needle [, int $offset = 0 ] ) |

내부함수 stripos()는 strpos()와 동일한 동작을 합니다. 차이점으로는 검색시 대소문자를 구분하지 않고 처리를 합니다.

**03.3.4 마지막 위치**

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

내부함수 strrpos() 는 문자열에서 검색하고자 하는 글자의 **마지막 위치의 값**을 정수 값으로 반환을 합니다.

**|내부함수|**

|  |
| --- |
| int **strrpos** ( string $haystack , string $needle [, int $offset = 0 ] ) |

**예제파일) strrpos.php**

|  |
| --- |
| <?php  $str = "abcdeghijkl**em-**abcdeghijkl**em-**1234567";  echo "원본 : " . $str . "<br>";  $a = **strrpos($str,"em-")**;  echo $a;  ?> |

**화면출력)**

원본 : abcdeghijklem-abcdeghijklem-1234567

25

**|내부함수|**

|  |
| --- |
| int **strripos** ( string $haystack , string $needle [, int $offset = 0 ] ) |

내부함수 strripos()는 strrpos() 함수와 동일하게 동작을 합니다. 차이점으로는 문자열에서 대소 문자를 구분하지 않고 문자열의 마지막 위치를 찾습니다.

**03.3.5 마스크 필터**

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

내부함수 strspn()는 문자열 $object의 첫 번째 문장에 대해서 마스크 **필터링된 길이**를 출력합니다.

**|내부함수|**

|  |
| --- |
| int **strspn** ( string $subject , string $mask [, int $start [, int $length ]] ) |

**예제파일) strspn.php**

|  |
| --- |
| <?php  // mask에 맞는 initial segment의 길이를 반환합니다.  $object = "**423336** is the answer to the 128th question.";  $mask = "1234567890abcdefhijslmnopqrstuvwxyz";  $var = strspn($object, $mask);  echo $var;  ?> |

**화면출력)**

6

내부함수 strcspn() 는 $subject 문자열에서 $mask에 포함되어 있지 않은 문자의 초기 세그먼트값의 길이를 출력합니다.

**|내부함수|**

|  |
| --- |
| int **strcspn** ( string $subject , string $mask [, int $start [, int $length ]] ) |

**예제파일) strcspn.php**

|  |
| --- |
| <?php  $a = strcspn('abcd', 'apple');  var\_dump($a); //0  $b = strcspn('abcd', 'banana');  var\_dump($b); // 0  $c = strcspn('hello', 'l');  var\_dump($c); // 2  $d = strcspn('hello', 'world');  var\_dump($d); // 2  $e = strcspn('abcdhelloabcd', 'abcd', -9);  var\_dump($e); //5  $f = strcspn('abcdhelloabcd', 'abcd', -9, -5);  var\_dump($f); //4  ?> |

**화면출력)**

int(0)

int(0)

int(2)

int(2)

int(5)

int(4)

**03.3.6 매칭검색**

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

내부함수 strpbrk()는 임의의 문자 집합에 대한 문자열 검색합니다. 매칭 검색된 이후의 문자열을 반환합니다.

**|내부함수|**

|  |
| --- |
| string **strpbrk** ( string $haystack , string $char\_list ) |

**예제파일) strpbrk.php**

|  |
| --- |
| <?php  $text = 'This is a Simple text.';  // "is is a Simple text." 를 출력합니다.  // 처음에 'i'가 먼저 매팅이 되었기 때문입니다.  echo strpbrk($text, 'mi');  echo "<br>";  // "Simple text."를 출력합니다.  echo strpbrk($text, 'S');  ?> |

**화면출력)**

is is a Simple text.

Simple text.

위의 실험은 임의의 문자들 $char\_list에 대해서 매칭 검색을 하여 처리합니다. 매칭되는 문자들은 한 개 또는 여러 개일 수 있습니다.

**03.4 문자열 비교**

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

문자열을 비교하여 처리하는 기능은 문자열을 처리중에서도 비중이 높은 활용도를 가지고 있는 부분들 입니다. PHP는 다양한 문자열 비교 함수들을 통하여 보다 쉽게 문자열 처리를 할 수 있습니다.

**03.4.1 문자열비교**

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

내부함수 strcmp()는 바이너리 형태로 두개의 문자열을 비교합니다. 두개의 문자열이 같을 경우 0보다 큰값을 반환합니다. 문자열을 비교할때는 대소문자를 같이 구별합니다.

**|내부함수|**

|  |
| --- |
| int **strcmp** ( string $str1 , string $str2 ) |

**예제파일) strcmp.php**

|  |
| --- |
| <?php  $str1 = "hello";  $str2 = "hello";  $str3 = "word";  if (strcmp($str1, $str2) == 0) {  echo $str1 ."== ". $str2 . "<br>";  } else {  echo $str1 ."!= ". $str2 . "<br>";  }  if (strcmp($str2, $str3) == 0 ) {  echo $str2 ."== ". $str3 . "<br>";  } else {  echo $str2 ."!= ". $str3 . "<br>";  }  ?> |

**화면출력)**

hello== hello

hello!= word

**|내부함수|**

|  |
| --- |
| int **strcoll** ( string $str1 , string $str2 ) |

내부함수 strcoll()은 현재 로케일 기반 문자열을 비교합니다. 현재 로케일이 C 또는 POSIX이면이 함수는 strcmp()와 같습니다.

**예제파일) strcoll.php**

|  |
| --- |
| <?php  $a = 'a';  $b = 'A';  print strcmp ($a, $b) . "<br>";  // prints 1  setlocale (LC\_COLLATE, 'C');  print "Locale based C: " . strcoll ($a, $b) . "<br>";  // prints 1  setlocale (LC\_COLLATE, 'de\_DE');  print "de\_DE: " . strcoll ($a, $b) . "<br>";  setlocale (LC\_COLLATE, 'de\_CH');  print "de\_CH: " . strcoll ($a, $b) . "<br>";  setlocale (LC\_COLLATE, 'en\_US');  print "en\_US: " . strcoll ($a, $b) . "<br>";  ?> |

**화면출력)**

1

Locale based C: 1

de\_DE: 1

de\_CH: 1

en\_US: 1

**03.4.2 문자열 Binary safe**

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

C언어와 같이 저수준의 언어의 경우 문자열은 메모리 세그먼트를 포인터로 표현을 합니다. 문자열의 마지막 종료기호로는 특수마크로 0바이트 또는 null 바이트를 사용합니다. 따라서 0과 같이 이러한 특수마크 기호는 문자열에 포함을 할 수 없습니다.

이러한 점으로 문자열을 저장하는 또다른 방법으로는 포인터와 문자열의 길이를 같이 저장을 하는 것입니다. 하지만 이러한 방법은 문자열을 관리하는데 두개의 값을 사용해야 되기 때문에 문자열을 처리하는 것은 매우 복잡합니다.

**|내부함수|**

|  |
| --- |
| int **strncmp** ( string $str1 , string $str2 , int $len ) |

내부함수 strncmp()는 첫 번째 n 문자의 Binary safe 문자열 비교합니다. 대소문자 구분합니다. str1가 str2보다 작은 경우 0보다 작은 값을 반환합니다. str2가 str1 보다 큰경우에는 >0 값을 반환합니다. 두개의 값이 같은 경우에는 0을 반환합니다.

**예제파일) strncmp.php**

|  |
| --- |
| <?php  echo strncmp("xybc","a3234",0);  // 0    echo "<br>";  echo strncmp("blah123","hohoho", 0);  //0  ?> |

**화면출력)**

0

0

**|내부함수|**

|  |
| --- |
| int **strcasecmp** ( string $str1 , string $str2 ) |

내부함수 strcasecmp()는 Binary safe 유형으로 대 / 소문자를 구분하지 않고 문자열을 비교 합니다.

**예제파일) strcasecmp.php**

|  |
| --- |
| <?php  $var1 = "Hello";  $var2 = "hello";  if (strcasecmp($var1, $var2) == 0) {  echo '$var1 와 $var2 은 대소문자를 구분하지 않고 동일한 문자열 입니다.';  }  ?> |

**화면출력)**

$var1 와 $var2 은 대소문자를 구분하지 않고 동일한 문자열 입니다.

**|내부함수|**

|  |
| --- |
| int **substr\_compare** ( string $main\_str , string $str , int $offset [, int $length [, bool $case\_insensitivity = false ]] ) |

내부함수 substr\_compare()는 바이너리 세이프 형태로 오프셋의 두 문자열 비교, 최대 길이 문자 비교합니다. substr\_compare ()는 오프셋 위치에서 main\_str을 str과 최대 길이 문자를 비교합니다.

**예제파일) substr\_compare.php**

|  |
| --- |
| <?php  echo substr\_compare("abcde", "bc", 1, 2); // 0  echo substr\_compare("abcde", "de", -2, 2); // 0  echo substr\_compare("abcde", "bcg", 1, 2); // 0  echo substr\_compare("abcde", "BC", 1, 2, true); // 0  echo substr\_compare("abcde", "bc", 1, 3); // 1  echo substr\_compare("abcde", "cd", 1, 2); // -1  echo substr\_compare("abcde", "abc", 5, 1); // warning  ?> |

**화면출력)**

0

0

0

0

1

-1

**03.4.3 자연순서**

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

자연순서 알고리즘을 통하여 문자열을 비교하고 처리할 수 있습니다.

**|내부함수|**

|  |
| --- |
| int **strnatcmp** ( string $str1 , string $str2 ) |

내부함수 strnatcmp()은 "자연 순서" 알고리즘을 사용하여 문자열을 비교 합니다.

**예제파일) strnatcmp.php**

|  |
| --- |
| <?php  $arr1 = $arr2 = array("img12.png", "img10.png", "img2.png", "img1.png");  echo "Standard string comparison\n";  usort($arr1, "strcmp");  print\_r($arr1);    echo "\nNatural order string comparison\n";  usort($arr2, "**strnatcmp**");  print\_r($arr2);  ?> |

**화면출력)**

Standard string comparison Array ( [0] => img1.png [1] => img10.png [2] => img12.png [3] => img2.png )

Natural order string comparison Array ( [0] => img1.png [1] => img2.png [2] => img10.png [3] => img12.png )

**|내부함수|**

|  |
| --- |
| int **strnatcasecmp** ( string $str1 , string $str2 ) |

내부함수 strnatcasecmp()은 "자연 순서" 알고리즘을 적용합니다. 대소 문자를 구분하지 않는 문자열 비교합니다.

**03.4.4 유사성**

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

문자열의 유사성을 비교하고 검사를 합니다.

**|내부함수|**

|  |
| --- |
| int **similar\_text** ( string $first , string $second [, float &$percent ] ) |

내부함수 similar\_text() 는 두개의 문자의 비슷한 정도를 계산합니다.

**예제파일) similar\_text.php**

|  |
| --- |
| <?php  $str1 = "안녕하세요! 지니 입니다.";  $str2 = "안녕하세요! jiny 입니다.";  similar\_text($str1, $str2, $percent);  echo "두 문장의 유사도는 $percent % 입니다.<br>";  similar\_text($str2, $str1, $percent);  echo "두 문장의 유사도는 $percent % 입니다.<br>";  ?> |

**화면출력)**

두 문장의 유사도는 84.848484848485 % 입니다.

두 문장의 유사도는 84.848484848485 % 입니다.

**|내부함수|**

|  |
| --- |
| string **soundex** ( string $str ) |

내부함수 soundex()는 문자열의 soundex 키값을 반환합니다.

**예제파일) soundex.php**

|  |
| --- |
| <?php  echo "Euler ".soundex("Euler") . " == " .  "Ellery ". soundex("Ellery") . "<br>";  echo "Gauss ".soundex("Gauss") . " == " .  "Ghosh ". soundex("Ghosh") . "<br>";  echo "Hilbert ".soundex("Hilbert") . " == " .  "Heilbronn ". soundex("Heilbronn") . "<br>";  ?> |

**화면출력)**

Euler E460 == Ellery E460

Gauss G200 == Ghosh G200

Hilbert H416 == Heilbronn H416

내부함수 levenshtein()은 두 문자열 사이의 Levenshtein 거리를 반환합니다. Levenshtein 거리는 문자열 str1을 str2로 변환하기 위해 교체하거나 삽입 또는 삭제해야하는 최소 문자 수를 말합니다.

**|내부함수|**

|  |
| --- |
| int **levenshtein** ( string $str1 , string $str2 ) |

**예제파일) levenshtein.php**

|  |
| --- |
| <?php  // 당근 : carrot 의 스펠링을 잘 못 입력합니다.  $input = 'carrrot';  // 단어 데이터들.  $words = array('apple','pineapple','banana','orange',  'radish','carrot','pea','bean','potato');  $shortest = -1;  // 입력한 단어를 비교합니다.  foreach ($words as $word) {  $lev = levenshtein($input, $word);  // 입력한 단어가 일치할때  if ($lev == 0) {  // 일치한 단어 설정  $closest = $word;  $shortest = 0;  // 단어가 일치하기 때문에, 반복문 종료  break;  }  if ($lev <= $shortest || $shortest < 0) {  $closest = $word;  $shortest = $lev;  }  }  echo "입력단어: $input <br>";  if ($shortest == 0) {  echo "정확한 단어: $closest <br>";  } else {  echo "혹시 원하는 단어가 $closest 입니까? <br>";  }  ?> |

**화면출력)**

입력단어: carrrot

혹시 원하는 단어가 carrot 입니까?

**|내부함수|**

|  |
| --- |
| string **metaphone** ( string $str [, int $phonemes = 0 ] ) |

내부함수 metaphone()은 문자열의 메타 폰 키를 계산합니다.

**예제파일) metaphone.php**

|  |
| --- |
| <?php  var\_dump(metaphone('programming'));  echo "<br>";  var\_dump(metaphone('programmer'));  echo "<br>";  var\_dump(metaphone('programming', 5));  echo "<br>";  var\_dump(metaphone('programmer', 5));  echo "<br>";  ?> |

**화면출력)**

string(7) "PRKRMNK"

string(6) "PRKRMR"

string(5) "PRKRM"

string(5) "PRKRM"

**03.5 문자열 치환**

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

치환이란 특정 문자열의 내용을 다른 문자열로 대체를 한다는 것입니다. 문자열 치환은 다양한 문자열 출력 결과물을 만들어 처리하는데 매우 유용한 기능입니다.

문자열의 내용을 치환하는 알고리즘은 복잡합니다. PHP는 문자열 치환 함수들을 통하여 보다 간단하게 처리를 할 수 있습니다.

**|내부함수|**

|  |
| --- |
| string **strtr** ( string $str , string $from , string $to ) |

내부함수 strtr()는 특정 문자열을 대체합니다. 참고로 문자열을 대치할 때는 문자열의 길이가 같아야 합니다. 만일 대체 하고자 하는 문자열의 길이가 더 클때는, 이후 문자열은 무시됩니다.

**예제파일) strtr.php**

|  |
| --- |
| <?php  $str = "안녕하세요. jiny 입니다.!";  echo $str."<br>";  $src = "jiny";  $dst = "hojinlee";  echo strtr($str, $src, $dst);  ?> |

**화면출력)**

안녕하세요. jiny 입니다.!

안녕하세요. hoji 입니다.!

**|내부함수|**

|  |
| --- |
| mixed **str\_replace** ( mixed $search , mixed $replace , mixed $subject [, int &$count ] ) |

내부함수 str\_replace()는 문자열 내의 특정 문자열을 검색하여 다른 문자열로 대체하여 줍니다.

**예제파일) str\_replace.php**

|  |
| --- |
| <?php  $string = "abcdefg";  $keyword = "cde";  $body = str\_replace($keyword, " 111111 ", $string);  echo $body;  ?> |

**화면출력)**

ab 111111 fg

**|내부함수|**

|  |
| --- |
| mixed **str\_ireplace** ( mixed $search , mixed $replace , mixed $subject [, int &$count ] ) |

내부함수 str\_ireplace()는 str\_replace ()의 대소 문자를 구별하지 않는 버전입니다.

**|내부함수|**

|  |
| --- |
| mixed **substr\_replace** ( mixed $string , mixed $replacement , mixed $start [, mixed $length ] ) |

내부함수 substr\_replace()는 문자열의 일부분 내에서 텍스트 바꾸기 합니다.

**예제파일) substr\_replace.php**

|  |
| --- |
| <?php  $var = 'ABCDEFGH:/MNRPQR/';  echo "원본: $var<hr/>\n";  /\* 전제 문자열을 'bob'으로 변경합니다. \*/  echo substr\_replace($var, 'bob', 0) . "<br />\n";  echo substr\_replace($var, 'bob', 0, strlen($var)) . "<br />\n";  /\* 문자열 앞에 'bob'을 추가합니다. \*/  echo substr\_replace($var, 'bob', 0, 0) . "<br />\n";  /\* 'MNRPQR' 부분을 'bob'으로 바꾸기 합니다. \*/  echo substr\_replace($var, 'bob', 10, -1) . "<br />\n";  echo substr\_replace($var, 'bob', -7, -1) . "<br />\n";  /\* 'MNRPQR' 부분을 삭제합니다. \*/  echo substr\_replace($var, '', 10, -1) . "<br />\n";  ?> |

**화면출력)**

원본: ABCDEFGH:/MNRPQR/bob

bob

bobABCDEFGH:/MNRPQR/

ABCDEFGH:/bob/

ABCDEFGH:/bob/

ABCDEFGH://

**03.6 문자**

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

문자는 문자열을 구성하는 요소 입니다. PHP는 이러한 문자들을 표현하고 처리를 할 수 있는 다양한 함수들을 제공합니다.

**|내부함수|**

|  |
| --- |
| string **chr** ( int $ascii ) |

내부함수 chr()는 아스키 코드값에 대한 문자를 출력합니다.

**예제파일) chr.php**

|  |
| --- |
| <?php  // 아스키 코드  echo "27 = ".chr(27)."<br>";  echo "65 = ".chr(65)."<br>";  echo "92 = ".chr(92)."<br>";  ?> |

**화면출력)**

27 =

65 = A

92 = \

**|내부함수|**

|  |
| --- |
| int **ord** ( string $string ) |

내부함수 ord()는 입력한 문자의 아스키 코드값을 출력합니다.

**예제파일) ord.php**

|  |
| --- |
| <?php  // 아스키 코드  echo "27 = ".chr(27)."<br>";  echo "65 = ".chr(65)."<br>";  echo "92 = ".chr(92)."<br>";  echo "<br>";  echo "A = ".ord('A')."<br>";  echo "+ = ".ord('+')."<br>";  echo "% = ".ord('%')."<br>";  ?> |

**화면출력)**

27 =

65 = A

92 = \

A = 65

+ = 43

% = 37

**|내부함수|**

|  |
| --- |
| string **str\_repeat** ( string $input , int $multiplier ) |

내부함수 str\_repeat()는 문자열을 지정한 횟수만큼 반복합니다.

**예제파일) str\_repeat.php**

|  |
| --- |
| <?php  echo str\_repeat("-=", 10);  ?> |

화면출력)

-=-=-=-=-=-=-=-=-=-=

**|내부함수|**

|  |
| --- |
| string **str\_pad** ( string $input , int $pad\_length [, string $pad\_string = " " [, int $pad\_type = STR\_PAD\_RIGHT ]] ) |

내부함수 str\_pad()는 문자열을 다른 문자열로 특정 길이를 채 웁니다.

**예제파일) str\_pad.php**

|  |
| --- |
| <?php  $input = "hojin";  echo str\_pad($input, 10) ."<br>";  echo str\_pad($input, 10, "-=", STR\_PAD\_LEFT) ."<br>";  echo str\_pad($input, 10, "\_", STR\_PAD\_BOTH) ."<br>";  echo str\_pad($input, 6, "\_\_\_") ."<br>";  echo str\_pad($input, 3, "\*") ."<br>";  ?> |

**화면출력)**

hojin

-=-=-hojin

\_\_hojin\_\_\_

hojin\_

hojin

**03.7 구분화**

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

구분화는 문자열을 특정한 규칙을 통하여 분리하는 기능입니다. 여러 개의 문자열 데이터를 특정키를 기준으로 연결되어 있을 경우 이를 처리하여 구분을 할 수 있습니다.

**03.7.1 토큰**

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

내부함수 strtok()는 문자열을 주어진 키로 토큰화 합니다.

|내부함수|

|  |
| --- |
| string **strtok** ( string $str , string $token ) |

**예제파일) strtok.php**

|  |
| --- |
| <?php  $str = "안녕하세요! 지니 PHP 코딩 입니다.";  // 공백 문자로 문자열을 토큰화 합니다.  $aaa = strtok($str," ");  $i=0;    while($aaa){  echo $i++ . "= ". $aaa . "<br>";  $aaa = strtok(" ");  }  ?> |

**화면출력)**

0= 안녕하세요!

1= 지니

2= PHP

3= 코딩

4= 입니다.

**|내부함수|**

|  |
| --- |
| array **explode** ( string $delimiter , string $string [, int $limit = PHP\_INT\_MAX ] ) |

내부함수 explode() 는 주어진 문자열을 구분자로 구분하여 배열로 변환합니다.

**|내부함수|**

|  |
| --- |
| string **implode** ( string $glue , array $pieces ) |

내부함수 implode()는 반대로 배열을 연결하여 문자열로 반환합니다.

**예제파일) implode.php**

|  |
| --- |
| <?php  $string = "aaa;bbb;ccc;ddd;eee";  $arr = explode(";", $string);  foreach ($arr as $key => $value) {  echo $key."=",$value."<br>";  }  $msg = implode(",", $arr);  echo $msg;  ?> |

**화면출력)**

0=aaa

1=bbb

2=ccc

3=ddd

4=eee

aaa,bbb,ccc,ddd,eee

**03.7.2 문자열 분리**

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

내부함수 chunk\_split()는 문자열을 비슷한 작은 크기 형태로 분리합니다. base64\_encode() 출력을 RFC 2045에 맞게 변환할 때 유용하게 사용할 수 있습니다. 모든 문자마다 종료 시퀀스 "\r\n"를 삽입합니다.

**|내부함수|**

|  |
| --- |
| string **chunk\_split** ( string $body [, int $chunklen = 76 [, string $end = "\r\n" ]] ) |

**예제파일) chunk\_split.php**

|  |
| --- |
| <?php  $str = "abcdefghijklem";  echo chunk\_split($str, 4) ."<br>";  ?> |

**화면출력)**

abcd efgh ijkl em

**|내부함수|**

|  |
| --- |
| array **str\_split** ( string $string [, int $split\_length = 1 ] ) |

내부함수 str\_split()은 문자열을 배열로 변환합니다.

**예제파일) str\_split.php**

|  |
| --- |
| <?php  $str = "Hello Friend";  // 한글자씩 배열로 변환 합니다.  $arr1 = str\_split($str);  print\_r($arr1);  echo "<br>";  // 3글자씩 배열로 변환 합니다.  $arr2 = str\_split($str, 3);  print\_r($arr2);  ?> |

**화면출력)**

Array ( [0] => H [1] => e [2] => l [3] => l [4] => o [5] => [6] => F [7] => r [8] => i [9] => e [10] => n [11] => d )

Array ( [0] => Hel [1] => lo [2] => Fri [3] => end )

**03.7.3 래핑**

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

내부함수 wordwrap()은 문자열을 주어진 문자 수로 랩핑합니다.

**|내부함수|**

|  |
| --- |
| string **wordwrap** ( string $str [, int $width = 75 [, string $break = "\n" [, bool $cut = false ]]] ) |

**예제파일) wordwrap.php**

|  |
| --- |
| <?php  $text = "The quick brown fox jumped over the lazy dog.";  $newtext = wordwrap($text, 20, "<br />\n");  echo $newtext;  ?> |

**화면출력)**

The quick brown fox

jumped over the lazy

dog.

**03.7.4 변수해서**

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

내부함수 parse\_str()은 입력된 하나의 문자열을 변수로 해석 합니다. 각각의 변수는 &로 구분하며, 변수명=값 형태로 지정을 할 수 있습니다.

**|내부함수|**

|  |
| --- |
| void **parse\_str** ( string $encoded\_string [, array &$result ] ) |

**예제파일) parse\_str.php**

|  |
| --- |
| <?php  $str = "name[]=jiny&name[]=lee&country=korea";  parse\_str($str);  echo $country . "<br>";  echo $name[0]."<br>";  echo $name[1]."<br>";  ?> |

**화면출력)**

korea

jiny

lee

**03.8 문자열 조작**

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

컴퓨터와 프로그램언어들은 초기 영문권을 중심으로 개발이 되면서 대부분의 언어처리들이 알파벳으로 많이 구성되어 있습니다. 알파벳은 특성상 대문자와 소문자로 구분이 됩니다.

PHP는 이러한 알파벳 문자의 특징들을 처리할 수 있는 함수들을 지원합니다.

**03.8.1 문자열 순서**

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

내부함수 strrev()는 입력된 문자열의 순서를 반대로 바꿉니다.

**|내부함수|**

|  |
| --- |
| string **strrev** ( string $string ) |

**예제파일) strrev.php**

|  |
| --- |
| <?php  $str = "abcdeghijklem-abcdeghijklem-1234567";  echo "원본 : " . $str . "<br>";  $a = strrev($str);  echo $a;  ?> |

**화면출력)**

원본 : abcdeghijklem-abcdeghijklem-1234567

7654321-melkjihgedcba-melkjihgedcba

**03.8.2 대소문자**

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

알파벳의 대소문자들을 변경할 수 있습니다.

**|내부함수|**

|  |
| --- |
| string **strtolower** ( string $string ) |

내부함수 strtolower(), mb\_strtolower() 는 알파벳 문자열을 소문자로 변경합니다.

**|내부함수|**

|  |
| --- |
| string **strtoupper** ( string $string ) |

내부함수 stroupper(), mb\_stroupper() 는 알파벳 문자열을 대문자로 변경합니다.

**예제파일) strtolower.php**

|  |
| --- |
| <?php  $lower = "ABCD";  echo $lower. "=" . strtolower($lower). "<br>";  $upper = "abcd";  echo $upper. "=" . strtoupper("abcd"). "<br>";  ?> |

**화면출력)**

ABCD=abcd

abcd=ABCD

**03.8.3 낙타표기**

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

알파벳의 단어나 문장들을 단어의 첫글자를 대문자로 표기하는 낙타표기법을 사용합니다. PHP는 단어의 낙타 표기법을 변환할 수 있는 함수들을 제공합니다.

**|내부함수|**

|  |
| --- |
| string **ucwords** ( string $str [, string $delimiters = " \t\r\n\f\v" ] ) |

내부함수 ucwords()는 단어를 낙타표기법 처럼 각각의 단어를 대문자로 변환합니다.

**|내부함수|**

|  |
| --- |
| string **ucfirst** ( string $str ) |

내부함수 ucfirst()는 전체 문자열 중에서 첫글자만 대문자로 변환합니다.

**예제파일) ucwords.php**

|  |
| --- |
| <?php  $string = "abcd ergh ijkl mnop";  // 첫 단어들의 대문자로 변경합니다.  echo ucwords($string) . "<br>";  // 문자열 전체에서 첫단어만 대문자로 변경합니다.  echo ucfirst($string) . "<br>";  ?> |

**화면출력)**

Abcd Ergh Ijkl Mnop

Abcd ergh ijkl mnop

**|내부함수|**

|  |
| --- |
| string **lcfirst** ( string $str ) |

내부함수 lcfirst()은 문자열의 첫 번째 문자를 소문자로 변환합니다.

**예제파일) lcfirst.php**

|  |
| --- |
| <?php  $str = 'HelloWorld';  $str = lcfirst($str);  echo $str;  ?> |

**화면출력)**

helloWorld

**03.9 변환**

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

정수형 자료의 경우 숫자의 값을 가지고 있습니다. 실수형의 자료의 경우도 실수의 숫자값을 가지고 있습니다. 하지만 정수형, 실수형의 표현은 문자로도 출력이 가능합니다.

정수값, 실수값을 문자로 정확하게 표기를 하기 위해서는 내부 숫자 변환 함수를 이용하면 편리합니다.

**03.9.1 숫자표기**

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

내부함수 number\_format()는 입력된 문자열을 기준으로 그룹화된 숫자 서식 형태로 변경을 할 수 있습니다. 함수의 입력 매개 변수는 1개, 2개, 4개로 전달합니다.

|내부함수|

|  |
| --- |
| string **number\_format** ( float $number [, int $decimals = 0 ] ) |

기본적으로 매개변수 1개만 입력되는 경우, 천단위 표기로 변경된 포맷을 출력합니다. 2번째 인자값은 소수점 자리수를 의미합니다.

3번째 인자와 4번째 인자는 같이 한쌍으로 입력을 하여야 합니다. 3번째는 소수점 표기 기호, 4번째는 천단위 표시 기호 입니다.

**예제파일) number\_format.php**

|  |
| --- |
| <?php  $number = 1234.56;  // 기본  // 매개인자 1개만 전달할 경우 천단위 구분자 쉼표(,)로 포맷변경 됩니다.  echo number\_format($number) ."<br>";  // 1,235  // 두번째 매개변수는 소수점 자리수  echo number\_format($number,5) ."<br>";  // 1,234.56000  // 3번째 인자는 = 소수점 표기 기호  // 4번재 인자는 = 천단위 표기 기호  echo number\_format($number, 2, ',', ' ') ."<br>";  // 1 234,56  $number = 1234.5678;  echo number\_format($number, 2, '.', ',');  // 1234.57  ?> |

**화면출력)**

1,235

1,234.56000

1 234,56

1,234.57

**03.9.2 통화표시**

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

경제학, 금융쪽에서 사용되는 숫자는 통화로 사용되는 경우가 많이 있습니다. 숫자를 통화로 표기하는 것은 각각의 나라마다 표현하고 처리하는 방법들이 다릅니다.

내부함수 money\_format()는 통화 형식의 문자열을 반환합니다. 참고로 윈도우즈 환경에서는 money\_format() 를 사용할 수 없습니다.

|내부함수|

|  |
| --- |
| string **money\_format** ( string $format , float $number ) |

**예제파일) money\_format.php**

|  |
| --- |
| <?php  $number = 1234.56;  // 로컬설정 en\_US  setlocale(LC\_MONETARY, 'en\_US');  echo "en\_US". money\_format('%i', $number) . "<br>";  // USD 1,234.56  // 이탈리아 포맷 2 decimals`  setlocale(LC\_MONETARY, 'it\_IT');  echo money\_format('%.2n', $number) . "\n";  // Eu 1.234,56  // 음수값  $number = -1234.525234;  // US 포맷  // 왼쪽 정밀도의 경우 10 자리  setlocale(LC\_MONETARY, 'en\_US');  echo money\_format('%(#10n', $number) . "\n";  // ($ 1,234.57)  // 위 형식과 비슷한 형식으로 2 자리의 오른쪽 자리 정밀도와  // '\*'를 채우기 문자로 추가합니다.  echo money\_format('%=\*(#10.2n', $number) . "\n";  // ($\*\*\*\*\*\*\*\*1,234.57)    // 왼쪽에서 14 자리의 너비, 8 자릿수의 왼쪽 자릿수, 2 자의 오른쪽 자릿수  // withouth 그룹화 문자 및 de\_DE 로켈의 국제 형식 사용을합시다.  setlocale(LC\_MONETARY, 'de\_DE');  echo money\_format('%=\*^-14#8.2i', 1234.56) . "\n";  // Eu 1234,56\*\*\*\*    // 전환 지정 전후에 몇 가지 문안을 추가해 보겠습니다.  setlocale(LC\_MONETARY, 'en\_GB');  $fmt = 'The final value is %i (after a 10%% discount)';  echo money\_format($fmt, 1234.56) . "\n";  // The final value is GBP 1,234.56 (after a 10% discount)  ?> |

**03.10 인코딩**

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

글로벌화된 소프트웨어의 개발로 인하여 다국어 처리와 다양한 문자 언어셋을 지원하는 것은 중요합니다. 문자열은 다양한 문자 언어셋으로 처리가 됩니다.

PHP는 문자열의 언어 인코딩을 변경을 할 수 있는 몇가지 함수들을 제공하고 있습니다.

**|내장함수|**

|  |
| --- |
| string **iconv** ( string $in\_charset , string $out\_charset , string $str ) |

내부함수 iconv()는 문자 인코딩을 변환 합니다.

**예제파일) iconv.php**

|  |
| --- |
| <?php  $text = "This is the Euro symbol '€'.";  echo 'Original : ', $text, "<br>";  echo 'TRANSLIT : ', iconv("UTF-8", "ISO-8859-1//TRANSLIT", $text), "<br>";  echo 'IGNORE : ', iconv("UTF-8", "ISO-8859-1//IGNORE", $text), "<br>";  ?> |

**화면출력)**

Original : This is the Euro symbol '€'.

TRANSLIT : This is the Euro symbol 'EUR'.

IGNORE : This is the Euro symbol ''.

**|내부함수|**

|  |
| --- |
| string **convert\_uuencode** ( string $data ) |

내부함수 convert\_uuencode()은 문자열을 uuencode 알고리즘으로 인코딩 합니다.

**예제파일) convert\_uuencode.php**

|  |
| --- |
| <?php  $string = "test\ntext text\r\n";  echo convert\_uuencode($string);  ?> |

**화면출력)**

0=&5S=`IT97AT('1E>'0-"@`` `

**|내부함수|**

|  |
| --- |
| string **convert\_uudecode** ( string $data ) |

내부함수 convert\_uudecode()는 uuencode 된 문자열을 디코딩합니다.

**예제파일) convert\_uudecode.php**

|  |
| --- |
| <?php  echo convert\_uudecode("+22!L;W9E(%!(4\"$`\n`");  ?> |

**화면출력)**

I love PHP!

**|내부함수|**

|  |
| --- |
| string **quoted\_printable\_encode** ( string $str ) |

내부함수 quoted\_printable\_encode()는 8 비트 문자열, 따옴표 붙은 인쇄 가능한 문자열로 변환합니다.

**|내부함수|**

|  |
| --- |
| string **quoted\_printable\_decode** ( string $str ) |

내부함수 quoted\_printable\_decode() 는 quoted-printable 문자열을 8비트 문자열로 변경을 합니다.

**|내부함수|**

|  |
| --- |
| string **convert\_cyr\_string** ( string $str , string $from , string $to ) |

내부함수 convert\_cyr\_string()는 하나의 Cyrillic 문자 집합을 다른 집합으로 변환 합니다.

**03.11 랜덤**

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

램덤이란 난수를 발행하는 알고리즘 입니다.

내부함수 random\_bytes()는 시스템을 통하여랜덤 바이트의 난수를 생성합니다.

**|내부함수|**

|  |
| --- |
| string **random\_bytes** ( int $length ) |

**예제파일) random\_bytes.php**

|  |
| --- |
| <?php  $bytes = random\_bytes(5);  var\_dump(bin2hex($bytes));  ?> |

**화면출력)**

string(10) "fab26daa1e"

**|내부함수|**

|  |
| --- |
| string **str\_shuffle** ( string $str ) |

내부함수 str\_shuffle()은 입력한 문자열을 무작위로 섞기를 합니다.

**예제파일) str\_shuffle.php**

|  |
| --- |
| <?php  $str = 'abcdef';  $shuffled = str\_shuffle($str);  echo $shuffled;  ?> |

**화면출력)**

efbcda

**03.12 해시 및 암호와**

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

문자열은 우리가 일반적으로 읽기 쉬운 형태의 문장들입니다. 이러한 문장들은 데이터를 처리하는데는 매우 편리합니다. 하지만, 데이터를 저장을 하거나 외부로 전송을 할 때 문장 그대로 처리를 하다보면 보안상 문제가 발생될 수 있습니다.

기본적으로 문자열을 보안처리 하는 방법으로는 암호화 하는 것입니다. PHP는 다양한 해시 및 암호화 모듈이 내장되어 있습니다. PHP 환경설정부분을 확인해보면 설치되어 있는 목록들을 보실 수 있습니다.

md2 md4 md5 sha1 sha224 sha256 sha384 sha512/224 sha512/256 sha512 sha3-224 sha3-256 sha3-384 sha3-512 ripemd128 ripemd160 ripemd256 ripemd320 whirlpool tiger128,3 tiger160,3 tiger192,3 tiger128,4 tiger160,4 tiger192,4 snefru snefru256 gost gost-crypto adler32 crc32 crc32b fnv132 fnv1a32 fnv164 fnv1a64 joaat haval128,3 haval160,3 haval192,3 haval224,3 haval256,3 haval128,4 haval160,4 haval192,4 haval224,4 haval256,4 haval128,5 haval160,5 haval192,5 haval224,5 haval256,5

**03.12.1 해쉬**

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

해쉬는 가장 기초적인 문자열 암호와 방식입니다.

**|내부함수|**

|  |
| --- |
| string **hash** ( string $algo , string $data [, bool $raw\_output = false ] ) |

내부함수 hash()는 해시 값을 생성합니다. 첫번째 인자 $algo는 알고리즘의 타입을 선택합니다. 예로 "md5", "sha256", "haval160,4" 등 두번째 인자인 $data 는 해시를 적용할 메시지 입니다. 세번째 인자인 $raw\_output 은 true 일때는 바이너리 형식, false는 소문자 hex로 반환합니다.

**예제파일) hash.php**

|  |
| --- |
| <?php  echo hash('ripemd160', 'hello world php!');  ?> |

화면출력)

271ab7cf2239daa049877e8c6fc73cfc8097d0de

**|내장함수|**

|  |
| --- |
| resource **hash\_init** ( string $algo [, int $options = 0 [, string $key = NULL ]] ) |

내장함수 hash\_init()는 추가된 문맥에 대해서 해시를 초기화 합니다.

* 옵션: HASH\_HMAC. 지정되면 키를 지정해야합니다.
* 키 : 옵션 HASH\_HMAC가 지정되면 HMAC 해시 메소드와 함께 사용할 공유 비밀 키를 매개 변수로 제공해야 합니다.

**예제파일) hash\_init.php**

|  |
| --- |
| <?php  $ctx = hash\_init('md5');  hash\_update($ctx, 'hello world ');  hash\_update($ctx, 'jinyPHP.');  echo hash\_final($ctx);  ?> |

**화면출력)**

c4009628bfac77c4060e51d14bd417a5

**|내부함수|**

|  |
| --- |
| array **hash\_algos** ( void ) |

내부함수 hash\_algos()는 등록된 해시 알고리즘 목록을 반환합니다.

**예제파일) hash\_algos.php**

|  |
| --- |
| <?php  print\_r(hash\_algos());  ?> |

**화면출력)**

Array ( [0] => md2 [1] => md4 [2] => **md5** [3] => **sha1** [4] => sha224 [5] => sha256 [6] => sha384 [7] => sha512/224 [8] => sha512/256 [9] => sha512 [10] => sha3-224 [11] => sha3-256 [12] => sha3-384 [13] => sha3-512 [14] => ripemd128 [15] => ripemd160 [16] => ripemd256 [17] => ripemd320 [18] => whirlpool [19] => tiger128,3 [20] => tiger160,3 [21] => tiger192,3 [22] => tiger128,4 [23] => tiger160,4 [24] => tiger192,4 [25] => snefru [26] => snefru256 [27] => gost [28] => gost-crypto [29] => adler32 [30] => crc32 [31] => crc32b [32] => fnv132 [33] => fnv1a32 [34] => fnv164 [35] => fnv1a64 [36] => joaat [37] => haval128,3 [38] => haval160,3 [39] => haval192,3 [40] => haval224,3 [41] => haval256,3 [42] => haval128,4 [43] => haval160,4 [44] => haval192,4 [45] => haval224,4 [46] => haval256,4 [47] => haval128,5 [48] => haval160,5 [49] => haval192,5 [50] => haval224,5 [51] => haval256,5 )

**03.12.2 MD5**

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

내부함수 md5()는 RFC1321 기준 md5 해시 알고리즘을 제공합니다.

**|내부함수|**

|  |
| --- |
| string **md5** ( string $str [, bool $raw\_output = false ] ) |

**예제파일) md5.php**

|  |
| --- |
| <?php  $password = "abcd1234";  echo "password = " . $password . "<br>";  echo "MD5 = " . md5($password) . "<br>";  echo "MD5 = " . md5($password) . "<br>";  echo "랜덤생성 예 === <br>";  echo "램덤 MD5 = " . md5(mt\_rand()) . "<br>";  ?> |

**화면출력)**

password = abcd1234

MD5 = e19d5cd5af0378da05f63f891c7467af

MD5 = e19d5cd5af0378da05f63f891c7467af

랜덤생성 예 ===

램덤 MD5 = bea084f3108d3fa7ce4f6826097e2cd8

**|내부함수|**

|  |
| --- |
| string **md5\_file** ( string $filename [, bool $raw\_output = false ] ) |

내부함수 md5\_file()은 주어진 파일에 대해서 MD5 해시값을 계산합니다.

**예제파일) md5\_file.php**

|  |
| --- |
| <?php  $file = 'md5\_file.php';  echo 'MD5 file hash of ' . $file . ': ' . md5\_file($file);  ?> |

**화면출력)**

MD5 file hash of md5\_file.php: **25adcad58baa2a626dfa53b98dff0995**

**03.12.3 crc32**

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

내부함수 crc32()는 문자열에 대해서 CRC32 polynomial 계산을 수행합니다.

**|내부함수|**

|  |
| --- |
| int **crc32** ( string $str ) |

CRC32는 보통 데이터 전송시 무결성 검증을 위해서 사용을 합니다. 입력된 문자열 스트링의 32비트 순환 중복 검사에 대한 결과를 출력합니다.

**예제파일) crc32.php**

|  |
| --- |
| <?php  $checksum = crc32("hello php world");  printf("%u\n", $checksum);  ?> |

**화면출력)**

2202403677

**03.12.4 sha1**

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

내부함수 sha1()은 문자열에 대해서 sha1 해시 계산을 처리합니다.

**|내부함수|**

|  |
| --- |
| string **sha1** ( string $str [, bool $raw\_output = false ] ) |

raw\_output이 true 인 경우에는 길이가 20 인 원시 바이너리 형식을 반환합니다. false 인 경우에는 40 문자 16 진수를 반환합니다.

**예제파일) sha1.php**

|  |
| --- |
| <?php  $str = 'apple';  // 40 문자 16 진수  echo $str ." = ". sha1($str). "<br>";  // 길이가 20 인 원시 바이너리 형식  echo $str ." = ". sha1($str,true). "<br>";  ?> |

화면출력)

apple = d0be2dc421be4fcd0172e5afceea3970e2f3d940

apple = о-�!�O�r����9p���@

**|내부함수|**

|  |
| --- |
| string **sha1\_file** ( string $filename [, bool $raw\_output = false ] ) |

내부함수 sha1\_file()은 주어진 파일에 대해서 SHA1 해시를 계산합니다.

**예제파일) sha1\_file.php**

|  |
| --- |
| <?php  foreach(glob('\*.exe') as $ent)  {  if(is\_dir($ent))  {  continue;  }  echo $ent . ' = (SHA1: ' . sha1\_file($ent) . ')' . "<br>";  }  ?> |

**화면출력)**

deplister.exe = (SHA1: 5aeb27623d25d042e101bb64ca011308cf2aa785)

php-cgi.exe = (SHA1: 5cdb18117c91de9db5616c8141456dff63dc4a75)

php-win.exe = (SHA1: 342aa529b6bf06b34e15ee7d3fef4dc87ee6199c)

php.exe = (SHA1: aeee36515446efd8ca4fccddc5e7b277f0fb217c)

phpdbg.exe = (SHA1: b1af4e81d2a146d9e3bd047c2a44b06b65999034)

**03.12.5 crypt**

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

내부함수 crypt() 함수는 표준 유닉스 DES 형태로 단방향 암호화된 문자열을 반환 합니다.

**|내부함수|**

|  |
| --- |
| string **crypt** ( string $str [, string $salt ] ) |

운영체제 별로 암호와 방식은 약간씩 다른데 MD5 로 대체하여 처리되기도 합니다. 암호화 작업시 기본 문자열 이외에 암호키(salt)를 사용할 수 있습니다.

**예제파일) crypt.php**

|  |
| --- |
| <?php    $password = "ABCD1234";  echo "암호 = " . $password . "<br>";  // salt 자동 생성  echo "DES 기반 암호 =" .crypt('ABCD1234') . "<br>";  // 사용자 salt  $salt = "복잡한 암호키 입니다.";  echo "DES 기반 암호 =" .crypt('ABCD1234',$salt) . "<br>";  ?> |

**화면출력)**

암호 = ABCD1234

DES 기반 암호 =$1$Lhg1VRxu$bLQyHdT/Cja/XbSgiNgGq.

DES 기반 암호 =��mEIN4PXgIk.

**03.12.6 str\_rot13**

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

내부함수 ROT13(Rotate by 13)은 카이사르 암호형식 입니다. 알파벳에 13을 밀어서 표기를 합니다. 즉 A 문자는 13을 더한 N 문자로 표기를 합니다. str\_rot13() 함수는 입력된 문자열에 대해서 ROT13 암호화 작업을 수행합니다.

|내부함수|

|  |
| --- |
| string **str\_rot13** ( string $str ) |

**예제파일) str\_rot13.php**

|  |
| --- |
| <?php  echo str\_rot13('PHP 4.3.0'); // CUC 4.3.0  ?> |

**화면출력)**

CUC 4.3.0

**03.13 문자열 출력**

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

PHP에서 변수의 데이터값을 출력할 수 있는 방법은 다양합니다. 간단하게 결과를 출력하는 echo 함수뿐만 아니라 다양한 포맷을 통하여 출력할 수 있는 몇 개의 함수들을 같이 제공합니다.

**03.13.1 출력**

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

내장함수 echo() 는 문자열을 출력합니다. php에서 가장 기본적이고 결과를 출력할 수 있는 방법입니다.

**|내부함수|**

|  |
| --- |
| void **echo** ( string $arg1 [, string $... ] ) |

**예제파일) echo.php**

|  |
| --- |
| <?php  echo "hello world";  ?> |

**화면출력)**

hello world

**|내부함수|**

|  |
| --- |
| int **print** ( string $arg ) |

내부함수 print() 함수는 echo()함수와 다르게 문자열을 화면에 출력후 성공여부를 논리값을 반환 합니다. 반환값은 화면의 정상적인 출력여부를 확인할 때 편리합니다.

**예제파일) print.php**

|  |
| --- |
| <?php  if(print("안녕하세요!")){  echo ">true";  } else {  echo ">false";  }  ?> |

**화면출력)**

안녕하세요!>true

**03.13.2 포맷출력**

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

포맷출력은 문자열을 그대로 출력하는 것이 아니라 포맷팅 처리를 하여 결과를 출력하는 방법입니다. 포맷팅 출력은 C언어 등에서 많이 이용하는 방법입니다. PHP에서도 C언어와 같이 다양한 포맷팅 처리 함수를 사용할 수 있습니다.

사용법 또한 매우 유사합니다. 사용되는 포맷 코드는 다음과 같습니다:

* %b : 바이너리 출력
* %c : 아스키문자 출력
* %d : 10진수 출력
* %f : 실수 출력
* %o : 8진수 출력
* %s : 문자열 출력
* %x : 16진수 소문자 출력
* %X : 16진수 대문자 출력

**|내부함수|**

|  |
| --- |
| int **printf** ( string $format [, mixed $args [, mixed $... ]] ) |

내부함수 printf()는 문자열을 지정한 포맷 방식으로 출력 합니다. 포맷은 출력 문자열에 지정한 타입형태로 데이터를 삽입하여 출력할 수 있는 기능입니다.

**예제파일) printf.php**

|  |
| --- |
| <?php  $name = "jiny";  if (printf("안녕하세요! %s 입니다.",$name)) {  echo ">true";  } else {  echo ">false";  }  ?> |

**화면출력)**

안녕하세요! jiny 입니다.>true

**|내부함수|**

|  |
| --- |
| int **fprintf** ( resource $handle , string $format [, mixed $args [, mixed $... ]] ) |

내장함수 fprintf()는 지정한 포맷을 스트림으로 출력합니다.

**예제파일) fprintf.php**

|  |
| --- |
| <?php  if (!($fp = fopen('date.txt', 'w'))) {  return;  }  // 지정한 포맷으로 파일 스트림에 출력합니다.  fprintf($fp, "%04d-%02d-%02d", $year, $month, $day);  ?> |

**|내부함수|**

|  |
| --- |
| int **vprintf** ( string $format , array $args ) |

내부함수 vprintf()는 형식화 된 문자열을 출력합니다.

**예제파일) vprintf.php**

|  |
| --- |
| <?php  vprintf("%04d-%02d-%02d", explode('-', '2017-8-6'));  ?> |

**화면출력)**

2017-08-06

**|내부함수|**

|  |
| --- |
| int **vfprintf** ( resource $handle , string $format , array $args ) |

내부함수 vfprintf()는 지정한 포맷을 스트림으로 출력합니다.

**예제파일) vfprintf.php**

|  |
| --- |
| <?php  if (!($fp = fopen('date.txt', 'w')))  return;  vfprintf($fp, "%04d-%02d-%02d", array($year, $month, $day));  ?> |

**|내부함수|**

|  |
| --- |
| string **sprintf** ( string $format [, mixed $args [, mixed $... ]] ) |

내부함수 sprintf() 는 printf() 함수와 달리 화면에 출력하지 않고 포맷형태로 변환하여 문자열을 반환합니다.

**예제파일) sprintf.php**

|  |
| --- |
| <?php  $name = "jiny";  $string = sprintf("안녕하세요! %s 입니다.",$name);  echo $string;  ?> |

**화면출력)**

안녕하세요! jiny 입니다.

**|내부함수|**

|  |
| --- |
| string **vsprintf** ( string $format , array $args ) |

내부함수 vsprintf()는 포맷 스트링을 반환합니다.

**예제파일) vsprintf.php**

|  |
| --- |
| <?php  echo vsprintf("%04d-%02d-%02d", explode('-', '2017-8-6'));  ?> |

**화면출력)**

2017-08-06

**03.13.3 포맷입력**

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

내부함수 sscanf()는 형식에 따라 입력된 문자열의 구문 분석합니다.

**|내부함수|**

|  |
| --- |
| mixed **sscanf** ( string $str , string $format [, mixed &$... ] ) |

**예제파일) sscanf.php**

|  |
| --- |
| <?php  // 시리얼 넘버를 읽어 옵니다.  list(**$serial**) = sscanf("SN/123456", "**SN/%d**");    $mandate = "july 06 2017";  list(**$month, $day, $year**) = sscanf($mandate, "**%s %d %d**");  echo "Item $serial was manufactured on: $year-" . substr($month, 0, 3) . "-$day\n";  ?> |

**화면출력)**

Item 123456 was manufactured on: 2017-jul-6

**03.14 html 문자열**

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

PHP는 웹사이트 개발 및 HTML을 처리하는데 매우 친화적으로 사용할 수 있는 언어 있습니다. HTML는 다양한 테그와 기호들을 포함하고 있습니다. PHP는 HTML 테그들을 처리하고 출력을 위한 다양한 함수들을 지원합니다.

**03.14.1 백슬래시**

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

문자열을 처리하는데 있어서 특수기호 백슬래시(\)는 SQL처리 및 문자열을 처리하는데 방해가 될 수 있습니다. 웹과 DB 연동을 하기위해서 문자열의 백슬래시 기호는 안전하게 처리를 해야 합니다. 이를 위해서 PHP는 백슬래시 처리에 관련된 함수들을 제공합니다.

**|내부함수|**

|  |
| --- |
| string **addslashes** ( string $str ) |

내부함수 addslashes()는 주어진 문자열에 백슬래시로 감싸 반환합니다. 백슬러시는 ‘ , “, \ 등 특수기호를 데이터베이스 쿼리등 작업을 할때 많이 사용을 합니다.

**예제파일) addslashes.php**

|  |
| --- |
| <?php  echo addslashes("c:\aaa\bbb\ccc");  echo "<br>";  echo addslashes("안녕하세요! 'jiny'님");  echo "<br>";  ?> |

**화면출력)**

c:\\aaa\\bbb\\ccc

안녕하세요! \'jiny\'님

**|내부함수|**

|  |
| --- |
| string **addcslashes** ( string $str , string $charlist ) |

내부함수 addcslashes()는 C 스타일로, 주어진 문자열에 백슬래시로 감싸 반환합니다.

**|내부함수|**

|  |
| --- |
| string **stripslashes** ( string $str ) |

내부함수 stripslashes()는 addslashes()함수의 반대입니다. 주어진 매개변수 문자열에서 quote 부분을 삭제하여 반환합니다.

**예제파일) stripslashes.php**

|  |
| --- |
| <?php  $str = "안녕하세요 '지니'입니다.";  $temp = addslashes($str);  echo $temp."<br>";  $temp2 = stripslashes($temp);  echo $temp2."<br>";  ?> |

**화면출력)**

안녕하세요 \'지니\'입니다.

안녕하세요 '지니'입니다.

**|내부함수|**

|  |
| --- |
| string **stripcslashes** ( string $str ) |

내부함수 stripcslashes() 는 addcslashes()의 반대 입니다.

**03.14.2 라인 브레이크**

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

콘솔 및 텍스트 기반의 문자열 처리에서 다음줄의 표시하는 기호는 \n을 사용합니다. 하지만 웹 화면에서는 다음줄(\n)이 반영이 되지 않습니다.

출력 결과를 웹으로 처리하기 위해서는 \n 기호를 웹에서의 다음줄인 <br>기호로 변경을 해야 합니다.

**|내부함수|**

|  |
| --- |
| string **nl2br** ( string $string [, bool $is\_xhtml = true ] ) |

내부함수 nl2br()는 HTML 라인브레이크로 <br>테그를 사용합니다.

**예제파일) nl2br.php**

|  |
| --- |
| <?php  echo nl2br("안녕하세요!\n 지니입니다.");  ?> |

**화면출력)**

안녕하세요!

지니입니다.

**03.14.3 테그제거**

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

내부함수 strip\_tags()는 문자열에서 HTML과 PHP 테그를 제거한 문자열을 반환 합니다.

**|내부함수|**

|  |
| --- |
| string **strip\_tags** ( string $str [, string $allowable\_tags ] ) |

**예제파일) strip\_tags.php**

|  |
| --- |
| <?php  $html = "  <h1>안녕하세요!</h1>  <br>  <?php phpinfo(); ?>  ";  echo $html;  echo "=============== <br>";    $temp = strip\_tags($html);  echo $temp;  ?> |

화면출력)

**안녕하세요!**

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

안녕하세요!

**03.14.4 html entities**

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

내부함수 htmlentities()은 변경이 가능한 모든 글자들을 HTML entities 코드로 변환합니다.

**|내부함수|**

|  |
| --- |
| string **htmlentities** ( string $string [, int $flags = ENT\_COMPAT | ENT\_HTML401 [, string $encoding = ini\_get("default\_charset") [, bool $double\_encode = true ]]] ) |

* ENT\_COMPAT 큰 따옴표만 변환합니다.
* ENT\_QUOTES 큰 따옴표와 작은 따옴표를 모두 변환합니다..
* ENT\_NOQUOTES 큰 따옴표와 작은 따옴표를 변환하지 않습니다.
* ENT\_IGNORE 빈 문자열을 반환하는 대신에 잘못된 코드 단위 시퀀스는 자동으로 무시합니다. 이 플래그를 사용하면 보안에 영향을 미칠 수 있으므로 사용하지 않는 것이 좋습니다.
* ENT\_SUBSTITUTE 잘못된 코드 단위 시퀀스를 유니 코드 대체 문자 U + FFFD (UTF-8) 또는 & # FFFD; (그렇지 않으면) 빈 문자열을 반환.
* ENT\_DISALLOWED 주어진 문서 유형에 대한 유효하지 않은 코드 포인트를 유니 코드 대체 문자 U + FFFD (UTF-8) 또는 & # FFFD; (그렇지 않은 경우) 그대로 남겨 둡니다.
* ENT\_HTML401 코드를 HTML 4.01로 처리.
* ENT\_XML1 코드를 XML 1로 처리.
* ENT\_XHTML 코드를 XHTML 로 처리.
* ENT\_HTML5 코드를 HTML 5로 처리.

**예제파일) htmlentities.php**

|  |
| --- |
| <?php  $str = "A 'quote' is <b>bold</b>";  // 출력: A 'quote' is &lt;b&gt;bold&lt;/b&gt;  echo htmlentities($str);  echo "<br>";  // 출력: A &#039;quote&#039; is &lt;b&gt;bold&lt;/b&gt;  echo htmlentities($str, ENT\_QUOTES);  echo "<br>";  $str = "\x8F!!!";  // 출력: an empty string  echo htmlentities($str, ENT\_QUOTES, "UTF-8");  echo "<br>";    // 출력: "!!!"  echo htmlentities($str, ENT\_QUOTES | ENT\_IGNORE, "UTF-8");  echo "<br>";  ?> |

**화면출력)**

'quote' is &lt;b&gt;bold&lt;/b&gt;

<br>

A &#039;quote&#039; is &lt;b&gt;bold&lt;/b&gt;

<br>

<br>

!!!

<br>

**|내부함수|**

|  |
| --- |
| string **html\_entity\_decode** ( string $string [, int $flags = ENT\_COMPAT | ENT\_HTML401 [, string $encoding = ini\_get("default\_charset") ]] ) |

내부함수html\_entity\_decode()은 HTML 엔터티를 문자로 변환합니다. html\_entity\_decode ()는 문자열의 모든 HTML 엔터티를 해당 문자로 변환한다는 점에서 htmlentities ()와 반대 함수입니다.

**예제파일) html\_entity\_decode.php**

|  |
| --- |
| <?php  $str = "I'll \"walk\" the <b>cat</b> now";  echo $str."<br>";  $a = htmlentities($str);  echo $a."<br>";  echo html\_entity\_decode($a);  ?> |

**화면출력)**

I'll "walk" the **cat** now

I'll "walk" the <b>cat</b> now

I'll "walk" the **cat** now

**|내부함수|**

|  |
| --- |
| string **htmlspecialchars** ( string $string [, int $flags = ENT\_COMPAT | ENT\_HTML401 [, string $encoding = ini\_get("default\_charset") [, bool $double\_encode = true ]]] ) |

내부함수 htmlspecialchars() 는 다음과 같은 HTML 특수문자를 다른 entity 코드로 변환하여 줍니다.

* & *&amp;*
* *"* *&quot;*,
* *'* *&#039;*
* *< &lt;*
* *> &gt;*

코드를 변환하여 출력하면 브라우저에서 코드를 분석하지 않고 html 코드 자체를 바로 출력할 수 있습니다.

**예제파일) htmlspecialchars.php**

|  |
| --- |
| <?php  $body = "<h1 class=\"aaa\" id='bb'>안녕하세요</h1> <br>";  echo $body;  echo htmlspecialchars($body);  ?> |

**화면출력)**

<h1 class="aaa" id='bb'>안녕하세요</h1> <br>

&lt;h1 class=&quot;aaa&quot; id='bb'&gt;안녕하세요&lt;/h1&gt; &lt;br&gt;

**|내부함수|**

|  |
| --- |
| string **htmlspecialchars\_decode** ( string $string [, int $flags = ENT\_COMPAT | ENT\_HTML401 ] ) |

내부함수 htmlspecialchars\_decode()는 특수 HTML 엔터티를 다시 문자로 변환합니다. htmlspecialchars() 함수의 반대함수 입니다.

**예제파일) htmlspecialchars\_decode.php**

|  |
| --- |
| <?php  $str = "<p>this -&gt; &quot;</p>\n";  echo htmlspecialchars\_decode($str);  echo htmlspecialchars\_decode($str, ENT\_NOQUOTES);  ?> |

**화면출력)**

<p>this -> "</p>

<p>this -> &quot;</p>

**|내부함수|**

|  |
| --- |
| array **get\_html\_translation\_table** ([ int $table = HTML\_SPECIALCHARS [, int $flags = ENT\_COMPAT | ENT\_HTML401 [, string $encoding = "UTF-8" ]]] ) |

내부함수 get\_html\_translation\_table()은 htmlspecialchars () 및 htmlentities ()에서 사용하는 변환 테이블을 반환합니다.

**예제파일) get\_html\_translation\_table.php**

|  |
| --- |
| <?php  var\_dump(get\_html\_translation\_table(HTML\_ENTITIES, ENT\_QUOTES | ENT\_HTML5));  ?> |

**화면출력)**

array(1511) {

[" "]=> string(5) " "

[" "]=> string(9) " "

["!"]=> string(6) "!"

["""]=> string(6) """

["#"]=> string(5) "#"

….. 중간생략

}

**03.14.5 메타**

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

내부함수 get\_meta\_tags()는 html 파일의 내용을 읽어 <head></head> 안에 설정되어 있는 메타 테그 값을 추출합니다. 추출된 모든 content를 배열형태로 반환합니다.

|내장함수|

|  |
| --- |
| array **get\_meta\_tags** ( string $filename [, bool $use\_include\_path = false ] ) |

**예제파일) get\_meta\_tags.php**

|  |
| --- |
| <?php  // html 파일  $tags = get\_meta\_tags("sample.htm");  echo "META author = ". $tags['author'] ."<br>";  echo "META keywords = ". $tags['keywords'] ."<br>";  echo "META description = ". $tags['description'] ."<br>";  ?> |

**화면출력)**

META author = jiny

META keywords = php documentation

META description = jinyPHP 셈플 html 파일입니다.

**|내부함수|**

|  |
| --- |
| string **quotemeta** ( string $str ) |

내부함수 quotemeta()는 매타 문자들에 대해서 백슬래시 가 붙어 있는 형태로 변환합니다.

* 메타문자: . \ + \* ? [ ^ ] ( $ )

**예제파일) quotemeta.php**

|  |
| --- |
| <?php  $str = "Hello world. (반가와요!)";  echo quotemeta($str);  ?> |

**화면출력)**

Hello world\. \(반가와요!\)

**03.14.6 escape**

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

내부함수 escapeshellcmd()는 이스케이프 쉘 메타 문자를 처리합니다.

**|내부함수|**

|  |
| --- |
| string **escapeshellcmd** ( string $command ) |

escapeshellcmd ()는 외부 입력값을 통하여 셸 명령을 실행할때 발생될 수 있는 악성 문자열등을 이스케이프 처리를 합니다. 악성문자열등은 시스템 명령을 생성하는 과정에 악의 적인 명령을 통하여 시스템 보안에 취약해질 수 있습니다.

exec() 또는 system() 함수를 실행전에 모든 입력 데이터들에 대해서 이스케이프 처리를 하는 것이 좋습니다.

다음 문자 앞에는 백 슬래시를 추가합니다. & # &`| \*? ~ <> ^ () [] {} $ \, \ x0A 및 \ xFF. Windows에서는 이러한 모든 문자와 % 및!가 대신 공백으로 대체됩니다.

**예제) escapeshellcmd.php**

|  |
| --- |
| <?php  $command = './configure '.$\_POST['options'];  $escaped\_command = escapeshellcmd($command);    system($escaped\_command);  ?> |

**|내부함수|**

|  |
| --- |
| string **escapeshellarg** ( string $arg ) |

내부함수 escapeshellarg()는 셸 인수로 사용되는 문자열을 이스케이프 처리합니다.

escapeshellarg ()는 문자열 주위에 작은 따옴표를 추가합니다. 또한 기존 작은 따옴표를 인용 / 이스케이프하여 문자열을 쉘 함수에 단일 안전한 인수로 전달 하도록 처리합니다.

셸 함수에는 exec (), system () 및 백틱 연산자가 포함됩니다.

Windows에서 escapeshellarg () 대신 백분율(%) 기호, 느낌표(!) 및 큰 따옴표를 공백으로 대체하고 문자열 주위에 큰 따옴표를 추가합니다.

**예제) escapeshellarg.php**

|  |
| --- |
| <?php  system('ls '.escapeshellarg($dir));  ?> |

**03.15 로케일 및 코드**

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

IntlChar는 PHP 7.x 으로 업그레이드 되면서 새롭게 추가된 클래스 입니다. 새로운 IntlChar 클래스는 추가 ICU 기능을 노출합니다. 유니 코드 문자를 조작하는 데 사용할 수 있습니다.

IntlChar클래스를 사용하기 위해서는 Intl 확장기능이 설치되어 있어야 합니다.

|  |
| --- |
| <?php  printf('%x', IntlChar::CODEPOINT\_MAX);  echo IntlChar::charName('@');  var\_dump(IntlChar::ispunct('!'));  ?> |

**화면출력)**

10ffff

COMMERCIAL AT

bool(true)

Unicode codepoint 이스케이프 구문은 16 진수 형식의 유니 코드 코드 포인트를 사용할 경우 코드 포인트를 UTF-8로 큰 따옴표로 묶인 문자열 또는 heredoc으로 출력합니다.

|  |
| --- |
| <?php  echo "\u{aa}";  echo "\u{0000aa}";  echo "\u{9999}";  ?> |

**|내부함수|**

|  |
| --- |
| string **setlocale** ( int $category , string $locale [, string $... ] ) |

내장함수 setlocale()은 로케일 정보를 설정합니다.

**|내부함수|**

|  |
| --- |
| array **localeconv** ( void ) |

내부함수 localeconv()은 숫자 형식 정보 가지고 옵니다.

**예제) localeconv.php**

|  |
| --- |
| <?php  setlocale(LC\_ALL, 'nl\_NL.UTF-8@euro');  $locale\_info = localeconv();  print\_r($locale\_info);  ?> |

**화면출력)**

Array ( [decimal\_point] => . [thousands\_sep] => [int\_curr\_symbol] => [currency\_symbol] => [mon\_decimal\_point] => [mon\_thousands\_sep] => [positive\_sign] => [negative\_sign] => [int\_frac\_digits] => 127 [frac\_digits] => 127 [p\_cs\_precedes] => 127 [p\_sep\_by\_space] => 127 [n\_cs\_precedes] => 127 [n\_sep\_by\_space] => 127 [p\_sign\_posn] => 127 [n\_sign\_posn] => 127 [grouping] => Array ( ) [mon\_grouping] => Array ( ) )

**|내부함수|**

|  |
| --- |
| string **nl\_langinfo** ( int $item ) |

내부함수 nl\_langinfo()는 쿼리 언어 및 로캘 정보. nl\_langinfo ()는 locale 카테고리의 개별 요소들을 액세스 하는데 이용합니다. 모든 요소를 ​​반환하는 localeconv ()와 달리 nl\_langinfo ()는 특정 요소만을 선택할 수 있습니다.