Chapter

03

문자열

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

PHP는 문자들을 다루고 처리할 수 있도록 다양한 문자열 함수를 제공하고 있습니다. 일부 내장 함수들은 추가 모듈 설치가 필요할 수도 있습니다.

03.1 문자열 변수

PHP는 C 언어 스타일의 인터프리터 언어입니다. 따라서 기본적인 문자열의 처리와 개념 또한 C 언어의 속성과 스타일을 따르는 경우가 많습니다.

PHP에서 문자열을 조작하고 처리하는 것은 생각보다 매우 쉽습니다. C 언어처럼 변수의 타입과 엄격한 문자와 문자열을 처리할 수 있는 규칙을 적용하지는 않습니다. 또한 PHP는 변수에 문자열을 입력함으로써 자동적으로 문자열이 변수형 타입으로 설정됩니다.

03.1.1 문자열 길이

문자열 데이터를 처리할 때 가장 많이 사용하는 기능은 문자열의 길이를 측정하는 것입니다. 문자열은 여러 개의 문자들의 집합이기 때문에 총 몇 개의 글자로 구성되어 있는지 확인하는 것이 중요합니다.

문자열의 길이를 기반으로 문자열을 처리해야 하는 패턴의 처리량이 결정됩니다. C 언어와 같은 언어에서는 문자열을 처리할 때 주로 배열을 통하여 저장합니다. 실제적으로 PHP 내부에서도 변수에 문자열을 저장하면 배열 형태로 처리할 수 있습니다.

$msg = “hello world!”;

PHP에서 위와 같이 변수의 생성과 문자열을 입력합니다. 변수 $msg는 문자열의 데이터를 가지고 있기 때문에 문자열 변수라고 할 수 있습니다. 그리고 $msg 변수는 배열의 접근 방식으로 한 글자 한 글자를 지정할 수 있습니다.

배열과 같은 특성을 가진 문자열 변수는 길이를 측정할 수 있습니다. 한 글자씩 문자를 포함하고 있는 메모리의 크기를 계산하면 되기 때문입니다. PHP에서는 문자열의 길이를 측정할 수 있는 내장 함수를 제공하고 있습니다.

| 내장 함수 |

```
int strlen ( string $string )
```

내장 함수 strlen()은 변수에 들어 있는 문자열의 길이를 확인할 수 있습니다. 함수의 매개변수 값으로 길이를 측정할 변수명을 입력하면 됩니다. 문자열의 길이는 정수 형태의 값으로 반환합니다. 문자열은 문자들의 연결된 배열의 크기를 측정하는 것과 같습니다.

예제 파일 | strlen-01.php

```
1  <?php
2      // 문자열 변수를 생성합니다.
3      $name = "Hello world!";
4
5      // 문자의 개수를 출력합니다.
```

```
6       echo "문자의 개수는:".strlen($name);
7
8   ?>
```

화면 출력

```
문자의 개수는: 12
```

문자열의 변수가 배열 형태를 띠고 있고, strlen() 함수를 통하여 문자열의 길이를 측정할 수 있었습니다. 이러한 특성을 이용하여 문자열 변수에서 한 글자씩 문자 값을 불러서 개별적으로 처리도 가능합니다.

예제 파일 | **strlen-02.php**

```
1   <?php
2       // 문자열 변수를 생성합니다.
3       $name = "Hello world!";
4
5       // 한 줄에 문자 한 글자씩 출력합니다.
6       for ($i=0;$i<strlen($name);$i++) {
7           echo $name[$i]."<br>";
8       }
9
10  ?>
```

화면 출력

```
H
e
l
l
o

w
o
r
l
d
!
```

위의 실험은 문자열의 변수를 배열로 접근하여 한 글자씩 출력합니다. 문자열을 배열 형

태로 접근할 때는 문자열 변수명 뒤에 대괄호를 통하여 접근합니다. 대괄호 안에 접근 값으로 0부터 시작되는 인덱스 값을 넣어주면 됩니다.

03.1.2 문자열 카운트

문자열 안에 또 다른 문자열을 포함하고 있을 경우 포함되고 있는 문자의 반복 개수를 측정할 수 있습니다.

| 내장 함수 |

```
int substr_count ( string $haystack , string $needle [, int $offset = 0 [, int $length ]] )
```

내장 함수 substr_count()은 원본의 $haystack 변수에서 검색하고자 하는 문자열 $needle 값이 몇 개를 포함하고 있는지를 확인합니다. 반환값은 정수입니다. 검색할 때는 대소문자를 구분하여 처리합니다.

예제 파일 | 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 두 번 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 건너뜁니다.
    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, 길이 제한  = ".substr_count($text, 'is', 3, 3) ."<br>";
```

```
18
19      // 오류 발생
20      // 오프셋 5 + 길이 10 = 총길이 15로 원본 $text 문자열 14보다 크기 때문에 오류 발생
21      echo substr_count($text, 'is', 5, 10) ."<br>";
22
23      // prints only 1, because it doesn't count overlapped substrings
24      $text2 = 'gcdgcdgcd';
25      echo substr_count($text2, 'gcdgcd') ."<br>";
26
27  ?>
```

화면 출력

```
This is a test
원본 문자열 길이 = 14
needle substring이 발생하는 횟수 =2
오프셋 적용 = 1
오프셋 적용, 길이 제한 = 0
```

위의 실험은 $text 문자열에서 is의 문자열을 포함하고 있는 문자열의 개수를 찾아서 처리합니다.

03.1.3 단어 개수 측정

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

| 내장 함수 |

```
mixed str_word_count ( string $string [, int $format = 0 [, string $charlist ]] )
```

내장 함수 str_word_count()은 입력된 문자열에서 단어의 개수를 찾아서 반환합니다.

예제 파일 | str_word_count.php

```
1   <?php
2       $str = "I love you";
```

```
3       echo str_word_count($str);
4
5   ?>
```

화면 출력

3

03.2 문자열 자르기

문자열을 처리하는 데 있어서 많이 사용하는 기능은 문자열을 분리하는 것입니다. 문자열은 다양한 형태로 존재하기 때문에 관련 함수들이 많이 지원됩니다.

03.2.1 문자열 제거

내장 함수 substr()은 입력한 문자열의 일부분을 잘라낼 수 있습니다. 문자열을 잘라낼 때는 문자열의 시작 위치와 끝 위치를 지정하여 해당 부분을 추출합니다.

| 내장 함수 |

```
string substr ( string $string , int $start [, int $length ] )
```

예제 파일 | substr.php

```
1   <?php
2       $string = "abcdefghijklmnopqrstuvwxyz";
3
4       // 1 위치 이후부터 표시함
5       echo substr($string,1)."<br>";
6
7       // 3 위치 이후부터 5개 표시함
8       echo substr($string,3,5)."<br>";
9
10  ?>
```

화면 출력

```
bcdefghijklmnopqrstuvwxyz
defgh
```

위의 예제는 문자열 $string 변수에서 특정 부분의 문자열을 추출합니다. 문자열을 추출할 때 세 번째 끝 값은 생략이 가능합니다. 끝 값을 생략할 때는 시작부터 끝까지를 모두 출력합니다.

03.2.2 공백 제거

문자열 변수는 공백도 문자로 취급을 합니다. 문자열을 가공하다가 보면 문자열 앞 또는 뒤에 쓸모가 없는 공백 문자들이 있는 경우가 있습니다. 이러한 경우 공백을 자동적으로 제거할 수 있는 내장 함수를 제공하고 있습니다.

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

| 내장 함수 | 양쪽 공백 제거

```
string trim ( string $str [, string $character_mask = " \t\n\r\0\x0B" ] )
```

내장 함수 trim()은 문자열의 앞뒤에 존재하는 공백 문자를 제거합니다.

| 내장 함수 | 우측 공백 제거

```
string rtrim ( string $str [, string $character_mask ] )
string chop (string str);
```

내장 함수 rtrim(), chop()은 문자열 오른쪽에 있는 공백 문자를 제거합니다. chop() 함수는 rtrim() 함수의 별칭입니다.

| 내장 함수 | 좌측 공백 제거

```
string ltrim ( string $str [, string $character_mask ] )
```

내장 함수 ltrim()은 문자열의 왼쪽에 있는 공백 문자를 제거합니다.

예제 파일 | trim.php

```
1  <?php
2     $string = "   jiny   ";
3     echo "안녕하세요!".$string."입니다.<br>";
4
5     // 앞뒤 공백 문자열을 삭제합니다.
6     echo "안녕하세요!".trim($string)."입니다.<br>";
7
8     // 오른쪽 공백을 제거합니다.
9     echo "안녕하세요!".chop($string)."입니다.<br>";
10    echo "안녕하세요!".rtrim($string)."입니다.<br>";
11
12    // 왼쪽 공백을 제거합니다.
13    echo "안녕하세요!".ltrim($string)."입니다.<br>";
14
15 ?>
```

화면 출력

```
안녕하세요! jiny 입니다.

안녕하세요!jiny입니다.
안녕하세요! jiny입니다.
안녕하세요! jiny입니다.
안녕하세요!jiny 입니다.
```

03.3 문자열 검색

문자열을 다양하게 처리하기 위해서는 문자열에서 특정 문자들의 위치를 찾아 처리하는 것입니다. 문자열의 내용을 검색하고 값을 반환하거나 위치 값을 구할 수 있습니다.

03.3.1 첫 번째 검색

내장 함수 strstr()은 문자열 안에서 찾을 문자의 첫 번째 위치를 구하고, 이후의 문자열을 데이터를 반환합니다. 문자를 검색할 때는 대소문자를 구별을 하는 점을 주의해야 합니다.

| 내장 함수 |

```
string strstr ( string $haystack , mixed $needle [, bool $before_needle = false ] )
```

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

예제 파일 | **strstr.php**

```
1  <?php
2     $str = "abcdeghijklem-abcdeghijklem-1234567";
3     echo "원본 : " . $str . "<br>";
4
5     // 문자열에서 em-으로 시작하는 위치를 찾아
6     // 이후의 문자열을 반환합니다.
7     $a = strstr($str,"em-");
8     echo $a;
9
10 ?>
```

화면 출력

```
원본 : abcdeghijklem-abcdeghijklem-1234567
em-abcdeghijklem-1234567
```

| 내장 함수 |

```
string stristr ( string $haystack , mixed $needle [, bool $before_needle = false ] )
```

내장 함수 stristr()은 strstr() 함수와 동일한 동작을 하는 함수입니다. 하지만 차이점으로는 대소문자를 구분하지 않습니다.

03.3.2 마지막 검색

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

| 내장 함수 |

```
string strrchr ( string $haystack , mixed $needle )
```

예제 파일 | strrchr.php

```
1  <?php
2     $str = "abcdeghijklem-abcdeghijklem-1234567";
3     echo strrchr($str,"em");
4
5  ?>
```

화면 출력

```
em-1234567
```

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

03.3.3 첫 번째 위치

내장 함수 strpos()는 문자열에서 검색하고자 하는 글자를 찾아 첫 번째 위치를 반환합니다. 만일 글자를 찾게 되면 문자열의 시작 위치를 정수값으로 반환합니다.

| 내장 함수 |

```
mixed strpos ( string $haystack , mixed $needle [, int $offset = 0 ] )
```

예제 파일 | **strpos.php**

```php
<?php
   $string = "abcdefg";
   $keyword = "cde";
   if (($pos = strpos($string, $keyword)) === false) {
      echo "Err] 찾는 문자열이 없습니다.";

   } else {
      echo "문자열 시작 위치 $pos 존재<br>";
   }

?>
```

화면 출력

문자열 시작 위치 2 존재

| 내장 함수 |

```
mixed stripos ( string $haystack , string $needle [, int $offset = 0 ] )
```

내장 함수 stripos()는 strpos()와 동일한 동작을 합니다. 차이점으로는 검색 시 대소문자를 구분하지 않고 처리합니다.

03.3.4 마지막 위치

내장 함수 strrpos()는 문자열에서 검색하고자 하는 글자의 **마지막 위치의 값**을 정수값으로 반환합니다.

| 내장 함수 |

```
int strrpos ( string $haystack , string $needle [, int $offset = 0 ] )
```

예제 파일 | **strrpos.php**

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

?>
```

화면 출력

```
원본 : abcdeghijklem-abcdeghijklem-1234567
25
```

| 내장 함수 |

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

내장 함수 strripos()는 strrpos() 함수와 동일하게 동작합니다. 차이점으로는 문자열에서 대소문자를 구분하지 않고 문자열의 마지막 위치를 찾습니다.

03.3.5 마스크 필터

내장 함수 strspn()은 문자열 $object의 첫 번째 문장에 대해서 마스크 **필터링된 길이**를 출력합니다.

| 내장 함수 |

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

예제 파일 | **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;

?>
```

화면 출력

6

내장 함수 strcspn()은 $subject 문자열에서 $mask에 포함되어 있지 않은 문자의 초기 세그먼트 값의 길이를 출력합니다.

| 내장 함수 |

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

예제 파일 | **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
?>
```

화면 출력

```
int(0)
int(0)
int(2)
int(2)
int(5)
int(4)
```

03.3.6 매칭 검색

내장 함수 strpbrk()는 임의의 문자 집합에 대한 문자열을 검색합니다. 매칭 검색된 이후의 문자열을 반환합니다.

| 내장 함수 |

```
string strpbrk ( string $haystack , string $char_list )
```

예제 파일 | strpbrk.php

```
<?php

    $text = 'This is a Simple text.';

    // "is is a Simple text."를 출력합니다.
    // 처음에 'i'가 먼저 매칭되었기 때문입니다.
    echo strpbrk($text, 'mi');
    echo "<br>";

    // "Simple text."를 출력합니다.
    echo strpbrk($text, 'S');

?>
```

화면 출력

```
is is a Simple text.
Simple text.
```

위의 예제는 임의의 문자 $char_list에 대해서 매칭 검색을 하여 처리합니다. 매칭되는 문자들은 1개 또는 여러 개일 수 있습니다.

03.4 문자열 비교

문자열을 비교하여 처리하는 기능은 문자열 처리 중에서도 비중이 높은 활용도가 있는 부분입니다. PHP는 다양한 문자열 비교 함수를 통하여 보다 쉽게 문자열을 처리할 수 있습니다.

03.4.1 문자열 비교

내장 함수 strcmp()는 바이너리 형태로 2개의 문자열을 비교합니다. 2개의 문자열이 같을 경우 0보다 큰 값을 반환합니다. 문자열을 비교할 때는 대소문자를 구별합니다.

| 내장 함수 |

```
int strcmp ( string $str1 , string $str2 )
```

예제 파일 | strcmp.php

```
1   <?php
2      $str1 = "hello";
3      $str2 = "hello";
4      $str3 = "word";
5
6      if (strcmp($str1, $str2) == 0) {
7         echo $str1 ."== ". $str2 . "<br>";
8      } else {
9         echo $str1 ."!= ". $str2 . "<br>";
10     }
11
12     if (strcmp($str2, $str3) == 0 ) {
13        echo $str2 ."== ". $str3 . "<br>";
```

```
    } else {
        echo $str2 ."!= ". $str3 . "<br>";
    }

?>
```

화면 출력

```
hello== hello
hello!= word
```

| 내장 함수 |

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

내장 함수 strcoll()은 현재 로케일 기반 문자열을 비교합니다. 현재 로케일이 C 또는 POSIX면 이 함수는 strcmp()와 같습니다.

예제 파일 | **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');
```

```
20      print "en_US: " . strcoll ($a, $b) . "<br>";
21
22  ?>
```

화면 출력

```
1
Locale based C: 1
de_DE: 1
de_CH: 1
en_US: 1
```

03.4.2 문자열 Binary safe

C 언어와 같이 저수준의 언어의 경우 문자열은 메모리 세그먼트를 포인터로 표현합니다. 문자열의 마지막 종료 기호로는 특수 마크로 0바이트 또는 null 바이트를 사용합니다. 따라서 0과 같이 이러한 특수 마크 기호는 문자열에 포함할 수 없습니다.

이러한 점으로 문자열을 저장하는 또 다른 방법으로는 포인터와 문자열의 길이를 함께 저장하는 것입니다. 하지만 이러한 방법은 문자열을 관리하는 데 2개의 값을 사용해야 하기 때문에 문자열을 처리하는 것은 매우 복잡합니다.

| 내장 함수 |

```
int strncmp ( string $str1 , string $str2 , int $len )
```

내장 함수 strncmp()는 첫 번째 n 문자의 Binary safe 문자열을 비교합니다. 대소문자 구분합니다. str1가 str2보다 작은 경우 0보다 작은 값을 반환합니다. str2가 str1보다 큰 경우에는 >0 값을 반환합니다. 2개의 값이 같은 경우에는 0을 반환합니다.

예제 파일 | strncmp.php

```
1   <?php
2       echo strncmp("xybc","a3234",0);
3       // 0
4
```

```
    echo "<br>";

    echo strncmp("blah123","hohoho", 0);
    //0

?>
```

화면 출력

```
0
0
```

| 내장 함수 |

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

내장 함수 strcasecmp()는 Binary safe 유형으로 대소문자를 구분하지 않고 문자열을 비교합니다.

예제 파일 | **strcasecmp.php**

```
<?php
    $var1 = "Hello";
    $var2 = "hello";
    if (strcasecmp($var1, $var2) == 0) {
        echo '$var1과 $var2는 대소문자를 구분하지 않고 동일한 문자열입니다.';
    }

?>
```

화면 출력

```
$var1 와 $var2 은 대소문자를 구분하지 않고 동일한 문자열입니다.
```

| 내장 함수 |

int **substr_compare** (string $main_str , string $str , int $offset [, int $length [, bool $case_insensitivity = false]])

내장 함수 substr_compare()는 바이너리 세이프 형태로 오프셋의 두 문자열 비교, 최대 길이 문자를 비교합니다. substr_compare()는 오프셋 위치에서 main_str을 str과 최대 길이 문자를 비교합니다.

예제 파일 | **substr_compare.php**

```
1   <?php
2      echo substr_compare("abcde", "bc", 1, 2); // 0
3      echo substr_compare("abcde", "de", -2, 2); // 0
4      echo substr_compare("abcde", "bcg", 1, 2); // 0
5      echo substr_compare("abcde", "BC", 1, 2, true); // 0
6      echo substr_compare("abcde", "bc", 1, 3); // 1
7      echo substr_compare("abcde", "cd", 1, 2); // -1
8      echo substr_compare("abcde", "abc", 5, 1); // warning
9
10  ?>
```

화면 출력

```
0
0
0
0
1
-1
```

03.4.3 자연 순서

자연 순서 알고리즘을 통하여 문자열을 비교하고 처리할 수 있습니다.

| 내장 함수 |

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

내장 함수 strnatcmp()는 '자연 순서' 알고리즘을 사용하여 문자열을 비교합니다.

예제 파일 | **strnatcmp.php**

```
1  <?php
2     $arr1 = $arr2 = array("img12.png", "img10.png", "img2.png", "img1.
      png");
3     echo "Standard string comparison\n";
4     usort($arr1, "strcmp");
5     print_r($arr1);
6
7     echo "\nNatural order string comparison\n";
8     usort($arr2, "strnatcmp");
9     print_r($arr2);
10 ?>
```

화면 출력

```
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 )
```

| 내장 함수 |

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

내장 함수 strnatcasecmp()는 '자연 순서' 알고리즘을 적용합니다. 대소문자를 구분하지 않는 문자열 비교합니다.

03.4.4 유사성

문자열의 유사성을 비교하고 검사합니다.

| 내장 함수 |

```
int similar_text ( string $first , string $second [, float &$percent ] )
```

내장 함수 similar_text()는 2개의 문자의 비슷한 정도를 계산합니다.

예제 파일 | 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>";

?>
```

화면 출력

```
두 문장의 유사도는 84.848484848485 % 입니다.
두 문장의 유사도는 84.848484848485 % 입니다.
```

| 내장 함수 |

```
string soundex ( string $str )
```

내장 함수 soundex()는 문자열의 soundex 키 값을 반환합니다.

예제 파일 | **soundex.php**

```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>";

?>
```

화면 출력

```
Euler E460 == Ellery E460
Gauss G200 == Ghosh G200
Hilbert H416 == Heilbronn H416
```

내장 함수 levenshtein()은 두 문자열 사이의 Levenshtein 거리를 반환합니다. Levenshtein 거리는 문자열 str1을 str2로 변환하기 위해 교체하거나 삽입 또는 삭제해야 하는 최소 문자 수를 말합니다.

| 내장 함수 |

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

예제 파일 | **levenshtein.php**

```php
<?php
   // 당근 : carrot의 스펠링을 잘못 입력합니다.
   $input = 'carrrot';

   // 단어 데이터들
   $words  = array('apple','pineapple','banana','orange',
             'radish','carrot','pea','bean','potato');

```

```php
    $shortest = -1;

    // 입력한 단어를 비교합니다.
    foreach ($words as $word) {

        $lev = levenshtein($input, $word);

        // 입력한 단어가 일치할 때
        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 "입력 단어: $input <br>";
    if ($shortest == 0) {
        echo "정확한 단어: $closest <br>";
    } else {
        echo "혹시 원하는 단어가 $closest 입니까? <br>";
    }

?>
```

화면 출력

```
입력 단어: carrrot
혹시 원하는 단어가 carrot 입니까?
```

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

내장 함수 metaphone()은 문자열의 메타폰 키를 계산합니다.

예제 파일 | **metaphone.php**

```
1  <?php
2      var_dump(metaphone('programming'));
3      echo "<br>";
4      var_dump(metaphone('programmer'));
5      echo "<br>";
6      var_dump(metaphone('programming', 5));
7      echo "<br>";
8      var_dump(metaphone('programmer', 5));
9      echo "<br>";
10
11 ?>
```

화면 출력

```
string(7) "PRKRMNK"
string(6) "PRKRMR"
string(5) "PRKRM"
string(5) "PRKRM"
```

03.5 문자열 치환

치환이란 특정 문자열의 내용을 다른 문자열로 대체한다는 것입니다. 문자열 치환은 다양한 문자열 출력 결과물을 만들어 처리하는 데 매우 유용한 기능입니다.

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

| 내장 함수 |

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

내장 함수 strtr()은 특정 문자열을 대체합니다. 참고로 문자열을 대체할 때는 문자열의 길이가 같아야 합니다. 만일 대체하고자 하는 문자열의 길이가 더 클 때는 이후 문자열은 무시됩니다.

예제 파일 | **strtr.php**

```
<?php
    $str = "안녕하세요. jiny 입니다.!";
    echo $str."<br>";

    $src = "jiny";
    $dst = "hojinlee";

    echo strtr($str, $src, $dst);

?>
```

화면 출력

```
안녕하세요. jiny 입니다.!
안녕하세요. hoji 입니다.!
```

| 내장 함수 |

mixed **str_replace** (mixed $search , mixed $replace , mixed $subject [, int &$count])

내장 함수 str_replace()는 문자열 내의 특정 문자열을 검색하여 다른 문자열로 대체합니다.

예제 파일 | **str_replace.php**

```
<?php
    $string = "abcdefg";
```

```
    $keyword = "cde";

    $body = str_replace($keyword, " 111111 ", $string);
    echo $body;

?>
```

화면 출력

```
ab 111111 fg
```

| 내장 함수 |

mixed **str_ireplace** (mixed $search , mixed $replace , mixed $subject [, int &$count])

내장 함수 str_ireplace()는 str_replace()의 대소문자를 구별하지 않는 버전입니다.

| 내장 함수 |

mixed **substr_replace** (mixed $string , mixed $replacement , mixed $start [, mixed $length])

내장 함수 substr_replace()는 문자열의 일부분 내에서 텍스트를 바꿉니다.

예제 파일 | **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. */
```

```
13      echo substr_replace($var, 'bob', 10, -1) . "<br />\n";
14      echo substr_replace($var, 'bob', -7, -1) . "<br />\n";
15
16      /* 'MNRPQR' 부분을 삭제합니다. */
17      echo substr_replace($var, '', 10, -1) . "<br />\n";
18
19  ?>
```

화면 출력

```
원본: ABCDEFGH:/MNRPQR/_____________________________________bob
bob
bobABCDEFGH:/MNRPQR/
ABCDEFGH:/bob/
ABCDEFGH:/bob/
ABCDEFGH://
```

03.6 문자

문자는 문자열을 구성하는 요소입니다. PHP는 이러한 문자들을 표현하고 처리할 수 있는 다양한 함수를 제공합니다.

| 내장 함수 |

```
string chr ( int $ascii )
```

내장 함수 chr()은 아스키 코드값에 대한 문자를 출력합니다.

예제 파일 | chr.php

```
1   <?php
2       // 아스키 코드
3       echo "27 = ".chr(27)."<br>";
4       echo "65 = ".chr(65)."<br>";
5       echo "92 = ".chr(92)."<br>";
6
```

```
7   ?>
```

화면 출력

```
27 =
65 = A
92 = \
```

| 내장 함수 |

```
int ord ( string $string )
```

내장 함수 ord()는 입력한 문자의 아스키 코드값을 출력합니다.

예제 파일 | ord.php

```
1   <?php
2       // 아스키 코드
3       echo "27 = ".chr(27)."<br>";
4       echo "65 = ".chr(65)."<br>";
5       echo "92 = ".chr(92)."<br>";
6
7       echo "<br>";
8
9       echo "A = ".ord('A')."<br>";
10      echo "+ = ".ord('+')."<br>";
11      echo "% = ".ord('%')."<br>";
12
13  ?>
```

화면 출력

```
27 =
65 = A
92 = \

A = 65
+ = 43
% = 37
```

| 내장 함수 |

string **str_repeat** (string $input , int $multiplier)

내장 함수 str_repeat()는 문자열을 지정한 횟수만큼 반복합니다.

예제 파일 | **str_repeat.php**

```
<?php
   echo str_repeat("-=", 10);

?>
```

화면 출력

```
-=-=-=-=-=-=-=-=-=-=
```

| 내장 함수 |

string **str_pad** (string $input , int $pad_length [, string $pad_string = " " [, int $pad_type = STR_PAD_RIGHT]])

내장 함수 str_pad()는 문자열을 다른 문자열로 특정 길이를 채웁니다.

예제 파일 | **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>";

?>
```

화면 출력

```
hojin
-=-=-hojin
__hojin___
hojin_
hojin
```

03.7 구분화

구분화는 문자열을 특정한 규칙을 통하여 분리하는 기능입니다. 여러 개의 문자열 데이터를 특정 키를 기준으로 연결되어 있을 경우 이를 처리하여 구분할 수 있습니다.

03.7.1 토큰

내장 함수 strtok()는 문자열을 주어진 키로 토큰화합니다.

| 내장 함수 |

```
string strtok ( string $str , string $token )
```

예제 파일 | strtok.php

```
<?php
   $str = "안녕하세요! 지니 PHP 코딩입니다.";

   // 공백 문자로 문자열을 토큰화합니다.
   $aaa = strtok($str," ");

   $i=0;

   while ($aaa) {
      echo $i++ . "= ". $aaa . "<br>";
      $aaa = strtok(" ");
   }

?>
```

화면 출력

```
0= 안녕하세요!
1= 지니
2= PHP
3= 코딩
4= 입니다.
```

| 내장 함수 |

```
array explode ( string $delimiter , string $string [, int $limit = PHP_INT_MAX ] )
```

내장 함수 explode()는 주어진 문자열을 구분자로 구분하여 배열로 변환합니다.

| 내장 함수 |

```
string implode ( string $glue , array $pieces )
```

내장 함수 implode()는 반대로 배열을 연결하여 문자열로 반환합니다.

예제 파일 | implode.php

```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;

?>
```

화면 출력

```
0=aaa
1=bbb
2=ccc
3=ddd
4=eee
aaa,bbb,ccc,ddd,eee
```

03.7.2 문자열 분리

내장 함수 chunk_split()는 문자열을 비슷한 작은 크기로 분리합니다. base64_encode() 출력을 RFC 2045에 맞게 변환할 때 유용하게 사용할 수 있습니다. 모든 문자마다 종료 시퀀스 "\r\n"을 삽입합니다.

| 내장 함수 |

```
string chunk_split ( string $body [, int $chunklen = 76 [, string $end = "\r\n" ]] )
```

예제 파일 | chunk_split.php

```
<?php
    $str = "abcdefghijklem";
    echo chunk_split($str, 4) ."<br>";

?>
```

화면 출력

```
abcd efgh ijkl em
```

| 내장 함수 |

```
array str_split ( string $string [, int $split_length = 1 ] )
```

내장 함수 str_split()는 문자열을 배열로 변환합니다.

예제 파일 | **str_split.php**

```php
<?php

    $str = "Hello Friend";

    // 한 글자씩 배열로 변환합니다.
    $arr1 = str_split($str);
    print_r($arr1);
    echo "<br>";

    // 세 글자씩 배열로 변환합니다.
    $arr2 = str_split($str, 3);
    print_r($arr2);

?>
```

화면 출력

```
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

내장 함수 wordwrap()은 문자열을 주어진 문자 수로 래핑합니다.

| 내장 함수 |

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

예제 파일 | **wordwrap.php**

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

```

```
    echo $newtext;

?>
```

화면 출력

```
The quick brown fox
jumped over the lazy
dog.
```

03.7.4 변수 해석

내장 함수 parse_str()은 입력된 하나의 문자열을 변수로 해석합니다. 각각의 변수는 &로 구분하며, 변수명=값 형태로 지정할 수 있습니다.

| 내장 함수 |

```
void parse_str ( string $encoded_string [, array &$result ] )
```

예제 파일 | 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>";

?>
```

화면 출력

```
korea
jiny
lee
```

03.8 문자열 조작

컴퓨터와 프로그램 언어들은 초기 영문권을 중심으로 개발이 되면서 대부분의 언어 처리는 알파벳으로 구성되어 있습니다. 알파벳은 특성상 대문자와 소문자로 구분됩니다.

PHP는 이러한 알파벳 문자의 특징을 처리할 수 있는 함수들을 지원합니다.

03.8.1 문자열 순서

내장 함수 strrev()는 입력된 문자열의 순서를 반대로 바꿉니다.

| 내장 함수 |

```
string strrev ( string $string )
```

예제 파일 | strrev.php

```
1   <?php
2      $str = "abcdeghijklem-abcdeghijklem-1234567";
3      echo "원본 : " . $str . "<br>";
4      $a = strrev($str);
5      echo $a;
6
7   ?>
```

화면 출력

```
원본 : abcdeghijklem-abcdeghijklem-1234567
7654321-melkjihgedcba-melkjihgedcba
```

03.8.2 대소문자

알파벳의 대소문자들을 변경할 수 있습니다.

| 내장 함수 |

string **strtolower** (string $string)

내장 함수 strtolower(), mb_strtolower()는 알파벳 문자열을 소문자로 변경합니다.

| 내장 함수 |

string **strtoupper** (string $string)

내장 함수 stroupper(), mb_stroupper()는 알파벳 문자열을 대문자로 변경합니다.

예제 파일 | **strtolower.php**

```
1  <?php
2     $lower = "ABCD";
3     echo $lower. "=" . strtolower($lower). "<br>";
4
5     $upper = "abcd";
6     echo $upper. "=" . strtoupper("abcd"). "<br>";
7
8  ?>
```

화면 출력

```
ABCD=abcd
abcd=ABCD
```

03.8.3 낙타 표기

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

| 내장 함수 |

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

내장 함수 ucwords()는 단어를 낙타 표기법처럼 각각의 단어를 대문자로 변환합니다.

| 내장 함수 |

```
string ucfirst ( string $str )
```

내장 함수 ucfirst()는 전체 문자열 중에서 첫 글자만 대문자로 변환합니다.

예제 파일 | **ucwords.php**

```
<?php
    $string = "abcd ergh ijkl mnop";

    // 첫 단어를 대문자로 변경합니다.
    echo ucwords($string) . "<br>";

    // 문자열 전체에서 첫 단어만 대문자로 변경합니다.
    echo ucfirst($string) . "<br>";

?>
```

화면 출력

```
Abcd Ergh Ijkl Mnop
Abcd ergh ijkl mnop
```

| 내장 함수 |

```
string lcfirst ( string $str )
```

내장 함수 lcfirst()는 문자열의 첫 번째 문자를 소문자로 변환합니다.

예제 파일 | lcfirst.php

```php
<?php
    $str = 'HelloWorld';
    $str = lcfirst($str);
    echo $str;

?>
```

화면 출력

```
helloWorld
```

03.9 변환

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

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

03.9.1 숫자 표기

내장 함수 number_format()은 입력된 문자열을 기준으로 그룹화된 숫자 서식 형태로 변경할 수 있습니다. 함수의 입력 매개변수는 1개, 2개, 4개로 전달합니다.

| 내장 함수 |

```
string number_format ( float $number [, int $decimals = 0 ] )
```

기본적으로 매개변수 1개만 입력되는 경우 천 단위 표기로 변경된 포맷을 출력합니다. 두 번째 인자값은 소수점 자리수를 의미합니다.

세 번째 인자와 네 번째 인자는 같이 한 쌍으로 입력해야 합니다. 세 번째는 소수점 표기

기호, 네 번째는 천 단위 표시 기호입니다.

예제 파일 | number_format.php

```php
<?php

    $number = 1234.56;

    // 기본
    // 매개인자 1개만 전달할 경우 천 단위 구분자 쉼표(,)로 포맷 변경됩니다.
    echo number_format($number) ."<br>";
    // 1,235

    // 두 번째 매개변수는 소수점 자리수
    echo number_format($number,5) ."<br>";
    // 1,234.56000

    // 세 번째 인자는 = 소수점 표기 기호
    // 네 번째 인자는 = 천 단위 표기 기호
    echo number_format($number, 2, ',', ' ') ."<br>";
    // 1 234,56

    $number = 1234.5678;
    echo number_format($number, 2, '.', ',');
    // 1234.57

?>
```

화면 출력

```
1,235
1,234.56000
1 234,56
1,234.57
```

03.9.2 통화 표시

경제학, 금융 쪽에서 사용되는 숫자는 통화로 사용되는 경우가 많습니다. 숫자를 통화로 표기하는 것은 각각의 나라마다 표현하고 처리하는 방법이 다릅니다.

내장 함수 money_format()은 통화 형식의 문자열을 반환합니다. 참고로 윈도우 환경에서는 money_format()을 사용할 수 없습니다.

| 내장 함수 |

string **money_format** (string $format , float $number)

예제 파일 | **money_format.php**

```php
<?php

    $number = 1234.56;

    // 로컬 설정 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

    // 음수 값
    $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 자리의 왼쪽 자릿수, 2 자리의 오른쪽 자릿수
```

```
    // withouth 그룹화 문자 및 de_DE 로케일의 국제 형식을 사용합시다.
    setlocale(LC_MONETARY, 'de_DE');
    echo money_format('%=*^-14#8.2i', 1234.56) . "\n";
    // Eu 1234,56****


    // 전환 지정 전후에 몇 가지 문안을 추가하겠습니다.
    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 처리됩니다.

PHP는 문자열의 언어 인코딩을 변경할 수 있는 몇 가지 함수를 제공하고 있습니다.

| 내장 함수 |

```
string iconv ( string $in_charset , string $out_charset , string $str )
```

내장 함수 iconv()는 문자 인코딩을 변환합니다.

예제 파일 | 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>";

7   ?>
```

화면 출력

```
Original : This is the Euro symbol '€'.
TRANSLIT : This is the Euro symbol 'EUR'.
IGNORE : This is the Euro symbol ''.
```

| 내장 함수 |

string **convert_uuencode** (string $data)

내장 함수 convert_uuencode()는 문자열을 uuencode 알고리즘으로 인코딩합니다.

예제 파일 | **convert_uuencode.php**

```
1   <?php
2       $string = "test\ntext text\r\n";
3       echo convert_uuencode($string);
4
5   ?>
```

화면 출력

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

| 내장 함수 |

string **convert_uudecode** (string $data)

내장 함수 convert_uudecode()는 uuencode된 문자열을 디코딩합니다.

예제 파일 | **convert_uudecode.php**

```
1   <?php
2       echo convert_uudecode("+22!L;W9E(%!(4\"$`\n`");
```

```
3
4   ?>
```

화면 출력

```
I love PHP!
```

| 내장 함수 |

```
string quoted_printable_encode ( string $str )
```

내장 함수 quoted_printable_encode()는 8비트 문자열, 따옴표 붙은 인쇄 가능한 문자열로 변환합니다.

| 내장 함수 |

```
string quoted_printable_decode ( string $str )
```

내장 함수 quoted_printable_decode()는 quoted-printable 문자열을 8비트 문자열로 변경합니다.

| 내장 함수 |

```
string convert_cyr_string ( string $str , string $from , string $to )
```

내장 함수 convert_cyr_string()은 하나의 Cyrillic 문자 집합을 다른 집합으로 변환합니다.

03.11 랜덤

랜덤이란 난수를 발행하는 알고리즘입니다.

내장 함수 random_bytes()는 시스템을 통하여 랜덤 바이트의 난수를 생성합니다.

| 내장 함수 |

string **random_bytes** (int $length)

예제 파일 | **random_bytes.php**

```
1  <?php
2     $bytes = random_bytes(5);
3     var_dump(bin2hex($bytes));
4
5  ?>
```

화면 출력

```
string(10) "fab26daa1e"
```

| 내장 함수 |

string **str_shuffle** (string $str)

내장 함수 str_shuffle()은 입력한 문자열을 무작위로 섞습니다.

예제 파일 | **str_shuffle.php**

```
1  <?php
2     $str = 'abcdef';
3     $shuffled = str_shuffle($str);
4
5     echo $shuffled;
6
7  ?>
```

화면 출력

```
efbcda
```

03.12 해시 및 암호화

문자열은 우리가 일반적으로 읽기 쉬운 형태의 문장들입니다. 이러한 문장들은 데이터를 처리하는 데는 매우 편리합니다. 하지만 데이터를 저장하거나 외부로 전송할 때 문장 그대로 처리하다 보면 보안상 문제가 발생할 수 있습니다.

기본적으로 문자열을 보안 처리하는 방법으로는 암호화가 있습니다. PHP는 다양한 해시 및 암호화 모듈이 내장되어 있습니다. PHP 환경 설정 부분을 확인해 보면 설치되어 있는 목록을 볼 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 해시

해시는 가장 기초적인 문자열 암호화 방식입니다.

| 내장 함수 |

```
string hash ( string $algo , string $data [, bool $raw_output = false ] )
```

내장 함수 hash()는 해시값을 생성합니다. 첫 번째 인자 $algo는 알고리즘의 타입을 선택합니다. 예로 'md5', 'sha256', 'haval160,4' 등 두 번째 인자인 $data는 해시를 적용할 메시지입니다. 세 번째 인자인 $raw_output은 true일 때는 바이너리 형식, false는 소문자 hex로 반환합니다.

예제 파일 | hash.php

```
1   <?php
```

```
    echo hash('ripemd160', 'hello world php!');

?>
```

화면 출력

```
271ab7cf2239daa049877e8c6fc73cfc8097d0de
```

| 내장 함수 |

```
resource hash_init ( string $algo [, int $options = 0 [, string $key = NULL ]] )
```

내장 함수 hash_init()는 추가된 문맥에 대해서 해시를 초기화합니다.

- 옵션: HASH_HMAC. 지정되면 키를 지정해야 합니다.
- 키: 옵션 HASH_HMAC가 지정되면 HMAC 해시 메서드와 함께 사용할 공유 비밀 키를 매개변수로 제공해야 합니다.

예제 파일 | hash_init.php

```
<?php
    $ctx = hash_init('md5');
    hash_update($ctx, 'hello world ');
    hash_update($ctx, 'jinyPHP.');
    echo hash_final($ctx);

?>
```

화면 출력

```
c4009628bfac77c4060e51d14bd417a5
```

| 내장 함수 |

```
array hash_algos ( void )
```

내장 함수 hash_algos()는 등록된 해시 알고리즘 목록을 반환합니다.

예제 파일 | **hash_algos.php**

```php
<?php
   print_r(hash_algos());

?>
```

화면 출력

```
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

내장 함수 md5()는 RFC1321 기준 md5 해시 알고리즘을 제공합니다.

| 내장 함수 |

```
string md5 ( string $str [, bool $raw_output = false ] )
```

예제 파일 | **md5.php**

```php
<?php
   $password = "abcd1234";
   echo "password = " . $password . "<br>";

   echo "MD5 = " . md5($password) . "<br>";
   echo "MD5 = " . md5($password) . "<br>";

```

```
8       echo "랜덤생성 예 === <br>";
9       echo "램덤 MD5 = " . md5(mt_rand()) . "<br>";
10
11  ?>
```

화면 출력

```
password = abcd1234
MD5 = e19d5cd5af0378da05f63f891c7467af
MD5 = e19d5cd5af0378da05f63f891c7467af
랜덤생성 예 ===
램덤 MD5 = bea084f3108d3fa7ce4f6826097e2cd8
```

| 내장 함수 |

string **md5_file** (string $filename [, bool $raw_output = false])

내장 함수 md5_file()은 주어진 파일에 대해서 MD5 해시값을 계산합니다.

예제 파일 | **md5_file.php**

```
1   <?php
2       $file = 'md5_file.php';
3       echo 'MD5 file hash of ' . $file . ': ' . md5_file($file);
4
5   ?>
```

화면 출력

```
MD5 file hash of md5_file.php: 25adcad58baa2a626dfa53b98dff0995
```

03.12.3 crc32

내장 함수 crc32()는 문자열에 대해서 CRC32 polynomial 계산을 수행합니다.

| 내장 함수 |

int **crc32** (string $str)

CRC32는 보통 데이터 전송 시 무결성 검증을 위해서 사용합니다. 입력된 문자열 스트링의 32비트 순환 중복 검사에 대한 결과를 출력합니다.

예제 파일 | crc32.php

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

?>
```

화면 출력

```
2202403677
```

03.12.4 sha1

내장 함수 sha1()은 문자열에 대해서 sha1 해시 계산을 처리합니다.

| 내장 함수 |

```
string sha1 ( string $str [, bool $raw_output = false ] )
```

raw_output이 true인 경우에는 길이가 20인 원시 바이너리 형식을 반환합니다. false인 경우에는 40문자 16진수를 반환합니다.

예제 파일 | sha1.php

```php
<?php
    $str = 'apple';

    // 40문자 16진수
    echo $str ." = ". sha1($str). "<br>";

    // 길이가 20인 원시 바이너리 형식
    echo $str ." = ". sha1($str,true). "<br>";
?>
```

화면 출력

```
apple = d0be2dc421be4fcd0172e5afceea3970e2f3d940
apple = o-◆!◆O◆r◆◆◆◆9p◆◆◆@
```

| 내장 함수 |

string **sha1_file** (string $filename [, bool $raw_output = false])

내장 함수 sha1_file()은 주어진 파일에 대해서 SHA1 해시를 계산합니다.

예제 파일 | **sha1_file.php**

```
<?php

   foreach(glob('*.exe') as $ent)
   {
      if (is_dir($ent))
      {
         continue;
      }

      echo $ent . ' = (SHA1: ' . sha1_file($ent) . ')' . "<br>";
   }

?>
```

화면 출력

```
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

내장 함수 crypt() 함수는 표준 유닉스 DES 형태로 단방향 암호화된 문자열을 반환합니다.

| 내장 함수 |

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

운영체제별로 암호화 방식은 약간씩 다른데, MD5로 대체하여 처리하기도 합니다. 암호화 작업 시 기본 문자열 이외에 암호키(salt)를 사용할 수 있습니다.

예제 파일 | **crypt.php**

```
1   <?php
2
3       $password = "ABCD1234";
4       echo "암호 = " . $password . "<br>";
5
6       // salt 자동 생성
7       echo "DES 기반 암호 =" .crypt('ABCD1234') . "<br>";
8
9       // 사용자 salt
10      $salt = "복잡한 암호키입니다.";
11      echo "DES 기반 암호 =" .crypt('ABCD1234',$salt) . "<br>";
12
13  ?>
```

화면 출력

```
암호 = ABCD1234
DES 기반 암호 =$1$Lhg1VRxu$bLQyHdT/Cja/XbSgiNgGq.
DES 기반 암호 =◆◆mEIN4PXgIk.
```

03.12.6 str_rot13

내장 함수 ROT13(Rotate by 13)은 카이사르 암호 형식입니다. 알파벳에 13을 밀어서 표기를 합니다. 즉 A 문자는 13을 더한 N 문자로 표기합니다. str_rot13() 함수는 입력된 문자열에 대해서 ROT13 암호화 작업을 수행합니다.

| 내장 함수 |

string **str_rot13** (string $str)

예제 파일 | str_rot13.php

```
1   <?php
2       echo str_rot13('PHP 4.3.0'); // CUC 4.3.0
3
4   ?>
```

화면 출력

```
CUC 4.3.0
```

03.13 문자열 출력

PHP에서 변수의 데이터 값을 출력할 수 있는 방법은 다양합니다. 간단하게 결과를 출력하는 echo 함수뿐만 아니라 다양한 포맷을 통하여 출력할 수 있는 몇 가지 함수를 제공합니다.

03.13.1 출력

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

| 내장 함수 |

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

예제 파일 | echo.php

```
1   <?php
```

```
2      echo "hello world";
3
4  ?>
```

화면 출력

```
hello world
```

| 내장 함수 |

```
int print ( string $arg )
```

내장 함수 print() 함수는 echo() 함수와 다르게 문자열을 화면에 출력 후 성공 여부를 논리값으로 반환합니다. 반환값은 화면의 정상적인 출력 여부를 확인할 때 편리합니다.

예제 파일 | print.php

```
1  <?php
2      if (print("안녕하세요!")) {
3          echo ">true";
4      } else {
5          echo ">false";
6      }
7
8  ?>
```

화면 출력

```
안녕하세요!>true
```

03.13.2 포맷 출력

포맷 출력은 문자열을 그대로 출력하는 것이 아니라 포맷팅 처리를 하여 결과를 출력하는 방법입니다. 포맷팅 출력은 C 언어 등에서 많이 이용하는 방법입니다. PHP에서도 C 언어와 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

| 내장 함수 |

```
int printf ( string $format [, mixed $args [, mixed $... ]] )
```

내장 함수 printf()는 문자열을 지정한 포맷 방식으로 출력합니다. 포맷은 출력 문자열에 지정한 타입 형태로 데이터를 삽입하여 출력할 수 있는 기능입니다.

예제 파일 | **printf.php**

```
1  <?php
2     $name = "jiny";
3
4     if (printf("안녕하세요! %s 입니다.",$name)) {
5        echo ">true";
6     } else {
7        echo ">false";
8     }
9
10 ?>
```

화면 출력

```
안녕하세요! jiny 입니다.>true
```

| 내장 함수 |

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

내장 함수 fprintf()는 지정한 포맷을 스트림으로 출력합니다.

예제 파일 | **fprintf.php**

```
<?php
   if (!($fp = fopen('date.txt', 'w'))) {
      return;
   }

   // 지정한 포맷으로 파일 스트림에 출력합니다.
   fprintf($fp, "%04d-%02d-%02d", $year, $month, $day);

?>
```

| 내장 함수 |

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

내장 함수 vprintf()는 형식화된 문자열을 출력합니다.

예제 파일 | **vprintf.php**

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

?>
```

화면 출력

```
2017-08-06
```

| 내장 함수 |

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

내장 함수 vfprintf()는 지정한 포맷을 스트림으로 출력합니다.

예제 파일 | **vfprintf.php**

```
<?php
   if (!($fp = fopen('date.txt', 'w')))
    return;

   vfprintf($fp, "%04d-%02d-%02d", array($year, $month, $day));

?>
```

| 내장 함수 |

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

내장 함수 sprintf()는 printf() 함수와 달리 화면에 출력하지 않고 포맷 형태로 변환하여 문자열을 반환합니다.

예제 파일 | **sprintf.php**

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

?>
```

화면 출력

안녕하세요! jiny 입니다.

| 내장 함수 |

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

내장 함수 vsprintf()는 포맷 스트링을 반환합니다.

예제 파일 | **vsprintf.php**

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

?>
```

화면 출력

```
2017-08-06
```

03.13.3 포맷 입력

내장 함수 sscanf()는 형식에 따라 입력된 문자열의 구문 분석합니다.

| 내장 함수 |

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

예제 파일 | **sscanf.php**

```
<?php
    // 시리얼 넘버를 읽어옵니다.
    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";

?>
```

화면 출력

```
Item 123456 was manufactured on: 2017-jul-6
```

03.14 html 문자열

PHP는 웹 사이트 개발 및 HTML을 처리하는 데 매우 친화적으로 사용할 수 있는 언어입니다. HTML는 다양한 태그와 기호를 포함하고 있습니다. PHP는 HTML 태그들을 처리하고 출력을 위한 다양한 함수를 지원합니다.

03.14.1 백슬래시

문자열을 처리하는 데 있어서 특수기호 백슬래시(\)는 SQL 처리 및 문자열을 처리하는 데 방해될 수 있습니다. 웹과 DB 연동을 위해서 문자열의 백슬래시 기호는 안전하게 처리해야 합니다. 이를 위해서 PHP는 백슬래시 처리에 관련된 함수를 제공합니다.

| 내장 함수 |

```
string addslashes ( string $str )
```

내장 함수 addslashes()는 주어진 문자열에 백슬래시로 감싸 반환합니다. 백슬래시는 ' , ", \ 등 특수기호를 데이터베이스 쿼리 등 작업할 때 많이 사용합니다.

예제 파일 | addslashes.php

```php
<?php
    echo addslashes("c:\aaa\bbb\ccc");
    echo "<br>";

    echo addslashes("안녕하세요! 'jiny'님");
    echo "<br>";

?>
```

화면 출력

```
c:\\aaa\\bbb\\ccc
안녕하세요! \'jiny\'님
```

| 내장 함수 |

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

내장 함수 addcslashes()는 C 스타일로 주어진 문자열에 백슬래시로 감싸 반환합니다.

| 내장 함수 |

string **stripslashes** (string $str)

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

예제 파일 | **stripslashes.php**

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

    $temp2 = stripslashes($temp);
    echo $temp2."<br>";

?>
```

화면 출력

```
안녕하세요 \'지니\'입니다.
안녕하세요 '지니'입니다.
```

| 내장 함수 |

string **stripcslashes** (string $str)

내장 함수 stripcslashes()는 addcslashes()의 반대입니다.

03.14.2 라인 브레이크

콘솔 및 텍스트 기반의 문자열 처리에서 다음 줄을 표시하는 기호는 \n을 사용합니다. 하지만 웹 화면에서는 다음 줄(\n)이 반영되지 않습니다.

출력 결과를 웹으로 처리하기 위해서는 \n 기호를 웹에서의 다음 줄인
 기호로 변경해야 합니다.

| 내장 함수 |

```
string nl2br ( string $string [, bool $is_xhtml = true ] )
```

내장 함수 nl2br()은 HTML 라인 브레이크로
 태그를 사용합니다.

예제 파일 | nl2br.php

```
1   <?php
2       echo nl2br("안녕하세요!\n 지니입니다.");
3
4   ?>
```

화면 출력

```
안녕하세요!
지니입니다.
```

03.14.3 태그 제거

내장 함수 strip_tags()는 문자열에서 HTML과 PHP 태그를 제거한 문자열을 반환합니다.

| 내장 함수 |

string **strip_tags** (string $str [, string $allowable_tags])

예제 파일 | **strip_tags.php**

```
<?php
    $html = "
    <h1>안녕하세요!</h1>
    <br>
    <?php phpinfo(); ?>
    ";

    echo $html;

    echo "=============== <br>";

    $temp = strip_tags($html);
    echo $temp;

?>
```

화면 출력

안녕하세요!

===============
안녕하세요!

03.14.4 html entities

내장 함수 htmlentities()는 변경 가능한 모든 글자들을 HTML entities 코드로 변환합니다.

string **htmlentities** (string $string [, int $flags = ENT_COMPAT | ENT_HTML401 [, string $encoding = ini_get("default_charset") [, bool $double_encode = true]]])

- ENT_COMPAT: 큰따옴표만 변환합니다.
- ENT_QUOTES: 큰따옴표와 작은따옴표를 모두 변환합니다.
- ENT_NOQUOTES: 큰따옴표와 작은따옴표를 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 반환합니다.
- 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 처리합니다.
- ENT_XML1: 코드를 XML 1로 처리합니다.
- ENT_XHTML: 코드를 XHTML로 처리합니다.
- ENT_HTML5: 코드를 HTML 5로 처리합니다.

예제 파일 | **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>";
```

```
11
12      $str = "\x8F!!!";
13
14      // 출력: an empty string
15      echo htmlentities($str, ENT_QUOTES, "UTF-8");
16      echo "<br>";
17
18      // 출력: "!!!"
19      echo htmlentities($str, ENT_QUOTES | ENT_IGNORE, "UTF-8");
20      echo "<br>";
21
22  ?>
```

화면 출력

```
 '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>
```

| 내장 함수 |

string **html_entity_decode** (string $string [, int $flags = ENT_COMPAT | ENT_HTML401 [, string $encoding = ini_get("default_charset")]])

내장 함수 html_entity_decode()는 HTML 엔터티를 문자로 변환합니다. html_entity_decode()는 문자열의 모든 HTML 엔터티를 해당 문자로 변환한다는 점에서 htmlentities()와 반대 함수입니다.

예제 파일 | **html_entity_decode.php**

```
1   <?php
2       $str = "I'll \"walk\" the <b>cat</b> now";
3       echo $str."<br>";
4
```

```php
    $a = htmlentities($str);
    echo $a."<br>";

    echo html_entity_decode($a);

?>
```

화면 출력

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

| 내장 함수 |

string **htmlspecialchars** (string $string [, int $flags = ENT_COMPAT | ENT_HTML401 [, string $encoding = ini_get("default_charset") [, bool $double_encode = true]]])

내장 함수 htmlspecialchars()는 다음과 같은 HTML 특수 문자를 다른 entity 코드로 변환해 줍니다.

- & &
- " ",
- ' '
- < <
- > >

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

예제 파일 | **htmlspecialchars.php**

```php
<?php
    $body = "<h1 class=\"aaa\" id='bb'>안녕하세요</h1> <br>";
    echo $body;

```

```
5       echo htmlspecialchars($body);
6
7   ?>
```

화면 출력

```
<h1 class="aaa" id='bb'>안녕하세요</h1> <br>
&lt;h1 class=&quot;aaa&quot; id='bb'&gt;안녕하세요&lt;/h1&gt; &lt;br&gt;
```

| 내장 함수 |

string **htmlspecialchars_decode** (string $string [, int $flags = ENT_COMPAT | ENT_HTML401])

내장 함수 htmlspecialchars_decode()는 특수 HTML 엔터티를 다시 문자로 변환합니다. htmlspecialchars() 함수의 반대 함수입니다.

예제 파일 | **htmlspecialchars_decode.php**

```
1   <?php
2       $str = "<p>this -&gt; &quot;</p>\n";
3
4       echo htmlspecialchars_decode($str);
5
6       echo htmlspecialchars_decode($str, ENT_NOQUOTES);
7
8   ?>
```

화면 출력

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

| 내장 함수 |

array **get_html_translation_table** ([int $table = HTML_SPECIALCHARS [, int $flags = ENT_COMPAT | ENT_HTML401 [, string $encoding = "UTF-8"]]])

내장 함수 get_html_translation_table()은 htmlspecialchars() 및 htmlentities()에서 사용하는 변환 테이블을 반환합니다.

예제 파일 | get_html_translation_table.php

```
<?php
    var_dump(get_html_translation_table(HTML_ENTITIES, ENT_QUOTES | ENT_
    HTML5));

?>
```

화면 출력

```
array(1511) {
["   "]=> string(5) "   "
[" "]=> string(9) " "
["!"]=> string(6) "!"
["""]=> string(6) """
["#"]=> string(5) "#"
….. 중간생략
}
```

03.14.5 메타

내장 함수 get_meta_tags()는 html 파일의 내용을 읽어 <head></head> 안에 설정되어 있는 메타 태그 값을 추출합니다. 추출된 모든 content를 배열 형태로 반환합니다.

| 내장 함수 |

array **get_meta_tags** (string $filename [, bool $use_include_path = false])

예제 파일 | get_meta_tags.php

```
<?php
    // html 파일
    $tags = get_meta_tags("sample.htm");

```

```
5       echo "META author = ". $tags['author'] ."<br>";
6       echo "META keywords = ". $tags['keywords'] ."<br>";
7       echo "META description = ". $tags['description'] ."<br>";
8
9   ?>
```

화면 출력

```
META author = jiny
META keywords = php documentation
META description = jinyPHP 샘플 html 파일입니다.
```

| 내장 함수 |

string **quotemeta** (string $str)

내장 함수 quotemeta()는 메타 문자들에 대해서 백슬래시가 붙어 있는 형태로 변환합니다.

- 메타 문자: . \ + * ? [^] ($)

예제 파일 | **quotemeta.php**

```
1   <?php
2       $str = "Hello world. (반가워요!)";
3       echo quotemeta($str);
4
5   ?>
```

화면 출력

```
Hello world\. \(반가워요!\)
```

03.14.6 escape

내장 함수 escapeshellcmd()는 이스케이프 셸 메타 문자를 처리합니다.

| 내장 함수 |

string **escapeshellcmd** (string $command)

escapeshellcmd()는 외부 입력값을 통하여 셸 명령을 실행할 때 발생할 수 있는 악성 문자열 등을 이스케이프 처리합니다. 악성 문자열 등은 시스템 명령을 생성하는 과정에 악의적인 명령을 통하여 시스템 보안을 취약하게 만들 수 있습니다.

exec() 또는 system() 함수를 실행 전에 모든 입력 데이터에 대해서 이스케이프 처리를 하는 것이 좋습니다.

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

예제 파일 | **escapeshellcmd.php**

```
1  <?php
2      $command = './configure '.$_POST['options'];
3
4      $escaped_command = escapeshellcmd($command);
5
6      system($escaped_command);
7
8  ?>
```

| 내장 함수 |

string **escapeshellarg** (string $arg)

내장 함수 escapeshellarg()는 셸 인수로 사용되는 문자열을 이스케이프 처리합니다.

escapeshellarg()는 문자열 주위에 작은따옴표를 추가합니다. 또한 기존 작은따옴표를 인용/이스케이프하여 문자열을 셸 함수에 단일 안전한 인수로 전달하도록 처리합니다.

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

윈도우에서 escapeshellarg() 대신 백분율(%) 기호, 느낌표(!) 및 큰따옴표를 공백으로 대체하고 문자열 주위에 큰따옴표를 추가합니다.

예제 파일 | escapeshellarg.php

```
1  <?php
2     system('ls '.escapeshellarg($dir));
3
4  ?>
```

03.15 로케일 및 코드

IntlChar는 PHP 7.x로 업그레이드되면서 새롭게 추가된 클래스입니다. 새로운 IntlChar 클래스는 추가 ICU 기능을 노출합니다. 이는 유니 코드 문자를 조작하는 데 사용할 수 있습니다. IntlChar 클래스를 사용하기 위해서는 Intl 확장 기능이 설치되어 있어야 합니다.

```
<?php
     printf('%x', IntlChar::CODEPOINT_MAX);
     echo IntlChar::charName('@');
     var_dump(IntlChar::ispunct('!'));
?>
```

| 내장 함수 |

```
string setlocale ( int $category , string $locale [, string $... ] )
```

내장 함수 setlocale()은 로케일 정보를 설정합니다.

| 내장 함수 |

array **localeconv** (void)

내장 함수 localeconv()는 숫자 형식 정보를 가져옵니다.

예제 파일 | **localeconv.php**

```
1  <?php
2     setlocale(LC_ALL, 'nl_NL.UTF-8@euro');
3
4     $locale_info = localeconv();
5     print_r($locale_info);
6
7  ?>
```

화면 출력

```
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 ( ) )
```

| 내장 함수 |

string **nl_langinfo** (int $item)

내장 함수 nl_langinfo()는 쿼리 언어 및 로케일 정보, nl_langinfo()는 locale 카테고리의 개별 요소들을 액세스하는 데 이용합니다. 모든 요소를 반환하는 localeconv()와 달리 nl_langinfo()는 특정 요소만을 선택할 수 있습니다.