jinyPHP 3권 - 05. 날짜

**05. 날짜**

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

서비스를 위한 응용프로그램을 개발하는 있어서 날짜와 시간정보는 매우 중요합니다. 대부분의 데이터들은 시간에 종속적인 의미를 가지는 경우가 많습니다. 날짜와 시간정보는 데이터를 처리하는데 있어서 뗄 수 없는 중요한 밀접관계를 가지고 있습니다.

이번장에서는 PHP에서 날짜와 시간에 관련된 함수들과 클래스를 살펴보도록 하겠습니다.

**05.1 실시간 환경 설정**

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

PHP가 설치되어 있는 지역과 환경에 따라서 기준이 다를 것입니다. 서버의 설정환경에 맞게 날짜와 시간을 계산할 필요가 있습니다. 특히 글로벌 서비스를 준비하고 있는 업체의 경우 날짜와 시간관련 처리에 대한 많은 고민이 필요로 합니다.

PHP는 날짜와 시간을 처리할 수 있는 다양한 내부함수들을 지원합니다. 하지만 날짜 시간관련 함수들은 php.ini설정값의 영향을 받습니다.

**05.2 날짜**

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

둥근 지구의 각 나라별로 날짜를 지정하고 처리하는 방법은 다양합니다. 이와 관련하여 국제 기준 시간과 날짜를 기준으로 계산을 합니다.

**|내부함수| 기준날짜**

|  |
| --- |
| string **gmdate** ( string $format [, int $timestamp = time() ] ) |

내부함수 gmdate() 는 CMT / CUT 기준의 날짜와 시간을 반환합니다. 기존 date() 함수와의 차이점은 기준날짜를 GMT 로 사용한다는 것입니다.

**예제파일) gmdate.php**

|  |
| --- |
| <?php  echo date("Y-m-d H:i:s");  echo "<br>";  echo "GMT(greenwich Mean Time) 기준 = ".gmdate("Y-m-d H:i:s");  ?> |

**화면출력)**

2017-06-18 06:31:56

GMT(greenwich Mean Time) 기준 = 2017-06-18 06:31:56

**|내부함수|**

|  |
| --- |
| array **date\_parse** ( string $date ) |

지정한 날짜에 대해서 상세정보를 배열로 반환합니다.

**예제파일) date\_parse.php**

|  |
| --- |
| <?php  print\_r(date\_parse("2017-08-07 10:00:00.5"));  ?> |

**화면출력)**

Array ( [year] => 2017 [month] => 8 [day] => 7 [hour] => 10 [minute] => 0 [second] => 0 [fraction] => 0.5 [warning\_count] => 0 [warnings] => Array ( ) [error\_count] => 0 [errors] => Array ( ) [is\_localtime] => )

**05.3 시간**

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

날짜와 같이 시간은 매우 중요합니다. 일반적인 시스템의 시/분//초 와 마이크로 초도 읽어 올 수 있습니다. 또한 타임스탬프를 지원합니다.

**05.3.1 시/분/초**

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

시스템을 통하여 현재의 시간을 알아내는 것은 시간처리를 작업을 하기 위한 첫단계 입니다. 내부함수 time()는 현재의 시간을 출력합니다.

**|내부함수| 현재시간**

|  |
| --- |
| int **time** ( void ) |

현재의 시간은 Unix epoch 이후의 시간을 timestamp 형식의 숫자 값으로 표기가 되는데 이를 가독성 있도록 표기를 하기 위해서는 date() 포맷함수를 이용하여 출력합니다.

**예제파일) time.php**

|  |
| --- |
| <?php  echo "현재의 시간은 = " . time() . "입니다. <br>";  // 두번째 인자로 주어진 시간을 포맷에 맞게 출력합니다.  echo date("H:i:s",time());  ?> |

**화면출력)**

현재의 시간은 = 1497766932입니다.

06:22:12

**|내부함수| 날짜와 시간**

|  |
| --- |
| array **getdate** ([ int $timestamp = time() ] ) |

내부함수 getdate()는 날짜와 시간 정보를 읽어 옵니다. 날짜와 시간의 반환 값은 배열로 받아 옵니다. 배열의 키를 통하여 각각의 값을 가지고 올 수 있습니다.

**예제파일) getdate.php**

|  |
| --- |
| <?php  $date = getdate();  echo "초 = ". $date['seconds'] . "<br>";  echo "분 = ". $date['minutes'] . "<br>";  echo "시 = ". $date['hours'] . "<br>";  echo "달날짜 = ". $date['mday'] . "<br>";  echo "요일(숫자) = ". $date['wday'] . "<br>";  echo "달(숫자) = ". $date['mon'] . "<br>";  echo "연도 = ". $date['year'] . "<br>";  echo "일년날짜수 = ". $date['yday'] . "<br>";  echo "요일 = ". $date['weekday'] . "<br>";  echo "달 = ". $date['month'] . "<br>";  ?> |

**화면출력)**

초 = 38

분 = 12

시 = 7

달날짜 = 18

요일(숫자) = 0

달(숫자) = 6

연도 = 2017

일년날짜수 = 168

요일 = Sunday

달 = June

**|내부함수| 현재 시간배열**

|  |
| --- |
| mixed **gettimeofday** ([ bool $return\_float = false ] ) |

내부함수 gettimeofday()는 현재의 시간을 배열 형태로 반환합니다. 배열의 키를 통하여 각각의 값에 접근을 할 수 있습니다.

**예제파일) gettimeofday.php**

|  |
| --- |
| <?php  $time = gettimeofday();  echo "현재 초 = " . $time['sec'] . "<br>";  echo "마이크로 초 = " . $time['usec'] . "<br>";  echo "서부 Greenwich 표준 = " . $time['minuteswest'] . "<br>";  echo "섬머타임 보정 = " . $time['dsttime'] . "<br>";  ?> |

**화면출력)**

현재 초 = 1497770514

마이크로 초 = 65156

서부 Greenwich 표준 = 0

섬머타임 보정 = 0

**|내부함수|**

|  |
| --- |
| string **gmstrftime** ( string $format [, int $timestamp = time() ] ) |

내부함수 gmstrftime()는 로컬 설정에 따른 GMT / UTC 날짜 , 시간 형식을 지정합니다.

**예제파일) gmstrftime.php**

|  |
| --- |
| <?php  setlocale(LC\_TIME, 'en\_US');  echo strftime("%b %d %Y %H:%M:%S", mktime(20, 0, 0, 12, 31, 98)) . "\n";  echo gmstrftime("%b %d %Y %H:%M:%S", mktime(20, 0, 0, 12, 31, 98)) . "\n";  ?> |

**화면출력)**

Dec 31 1998 20:00:00 Dec 31 1998 20:00:00

**|내부함수|**

|  |
| --- |
| int **idate** ( string $format [, int $timestamp = time() ] ) |

내부함수 idate()는 정수로 현지 시간 / 날짜 형식 지정

**예제파일) idate.php**

|  |
| --- |
| <?php  $timestamp = strtotime('1st January 2017');  // this prints the year in a two digit format  // however, as this would start with a "0", it  // only prints "4"  echo idate('y', $timestamp);  ?> |

**화면출력)**

17

**|내부함수|**

|  |
| --- |
| array **localtime** ([ int $timestamp = time() [, bool $is\_associative = false ]] ) |

내부함수 localtime()는 현지 시간을 반환합니다

**예제파일) localtime.php**

|  |
| --- |
| <?php  $localtime = localtime();  $localtime\_assoc = localtime(time(), true);  print\_r($localtime);  print\_r($localtime\_assoc);  ?> |

**화면출력)**

Array ( [0] => 24 [1] => 6 [2] => 9 [3] => 7 [4] => 7 [5] => 117 [6] => 1 [7] => 218 [8] => 0 ) Array ( [tm\_sec] => 24 [tm\_min] => 6 [tm\_hour] => 9 [tm\_mday] => 7 [tm\_mon] => 7 [tm\_year] => 117 [tm\_wday] => 1 [tm\_yday] => 218 [tm\_isdst] => 0 )

**05.3.2 마이크로초**

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

시/분/초 외에도 정밀한 마이크로초 단위의 시간을 읽어 올 수 있습니다.

마이크로초는 프로그램의 실행속도등을 측정할 때 자주 사용하는 시간함수 입니다. 매우 정밀한 시간을 표시하기 때문에 프로그램 시작전에 한번 실행을 하고, 프로그램 마지막에 시간을 한번 더 측정하여 프로그램의 동작시간을 측정할 수 있습니다.

**|내부함수| 마이크로초**

|  |
| --- |
| mixed **microtime** ([ bool $get\_as\_float = false ] ) |

내장함수 microtime()는 현재의 1/1000 초 단위로 출력합니다.

**예제파일) microtime.php**

|  |
| --- |
| <?php  $mtime = microtime();  echo $mtime;  ?> |

**화면출력)**

0.14970800 1497771297

**05.3.3 타임스탬프**

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

시간을 타임스탬프 형태로 사용을 할 수 있습니다. 타임 스탬프는 특정한 시각을 표현하는 문자열 입니다. 두개 이상의 시간을 비교하거나 계산을 하기 위해서 자주 사용을 합니다.

**|내부함수| 타임스템프 생성**

|  |
| --- |
| int **gmmktime** ([ int $hour = gmdate("H") [, int $minute = gmdate("i") [, int $second = gmdate("s") [, int $month = gmdate("n") [, int $day = gmdate("j") [, int $year = gmdate("Y")]]]]]] ) |

내부함수 gmmktime()는 GMT 기준으로 타임스탬프를 생성합니다.

**예제파일) gmmktime.php**

|  |
| --- |
| <?php  // Prints: jun 18, 2017  echo "jun 18, 2017 is on a " . date("l", gmmktime(0, 0, 0, 6, 18, 2017));  ?> |

**화면출력)**

jun 18, 2017 is on a Sunday

**|내부함수| 지정날짜**

|  |
| --- |
| int **mktime** ([ int $hour = date("H") [, int $minute = date("i") [, int $second = date("s") [, int $month = date("n") [, int $day = date("j") [, int $year = date("Y") ]]]]]] ) |

내장함수 mktime()는 지정된 날짜의 타임스템프를 생성합니다.

**예제파일) mktime.php**

|  |
| --- |
| <?php  $hour = date("H");  $minute = date("i");  $second = date("s");  $month = date("n");  $day = date("d");  $year = date("Y");  $timeStamp = mktime($hour, $minute, $second, $month, $day, $year);  // 두번째 인자로 주어진 시간을 포맷에 맞게 출력합니다.  echo date("Y-m-d H:i:s",$timeStamp);  ?> |

**화면출력)**

2017-06-18 07:32:17

**|내부함수|**

|  |
| --- |
| int **strtotime** ( string $time [, int $now = time() ] ) |

영어 텍스트 datetime을 Unix 타임 스탬프로 구문 분석합니다.

**예제파일) strtotime.php**

|  |
| --- |
| <?php  echo strtotime("now"), "<br>";  echo strtotime("10 September 2000"), "<br>";  echo strtotime("+1 day"), "<br>";  echo strtotime("+1 week"), "<br>";  echo strtotime("+1 week 2 days 4 hours 2 seconds"), "<br>";  echo strtotime("next Thursday"), "<br>";  echo strtotime("last Monday"), "<br>";  ?> |

**화면출력)**

1502097067

968544000

1502183467

1502701867

1502889069

1502323200

1501459200

**05.3.4 일출/일몰**

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

내부함수 date\_sun\_info()는 일몰(sunset) / 일출(sunrise) 및 황혼(twilight) 시작 / 끝에 대한 정보를 읽어 옵니다.

**|내부함수|**

|  |
| --- |
| array **date\_sun\_info** ( int $time , float $latitude , float $longitude ) |

**예제파일) date\_sun\_info.php**

|  |
| --- |
| <?php  $sun\_info = date\_sun\_info(strtotime("2017-08-07"), 31.7667, 35.2333);  foreach ($sun\_info as $key => $val) {  echo "$key: " . date("H:i:s", $val) . "\n";  }  ?> |

**화면출력)**

sunrise: 02:59:11 sunset: 16:30:21 transit: 09:44:46 civil\_twilight\_begin: 02:33:08 civil\_twilight\_end: 16:56:24 nautical\_twilight\_begin: 02:01:51 nautical\_twilight\_end: 17:27:41 astronomical\_twilight\_begin: 01:29:07 astronomical\_twilight\_end: 18:00:25

**|내부함수|**

|  |
| --- |
| mixed **date\_sunrise** ( int $timestamp [, int $format = SUNFUNCS\_RET\_STRING [, float $latitude = ini\_get("date.default\_latitude") [, float $longitude = ini\_get("date.default\_longitude") [, float $zenith = ini\_get("date.sunrise\_zenith") [, float $gmt\_offset = 0 ]]]]] ) |

내부함수 date\_sunrise()는 위치와 날짜에 대한 일출 시간을 계산합니다.

**예제파일) date\_sunrise.php**

|  |
| --- |
| <?php  /\* calculate the sunrise time for Lisbon, Portugal  Latitude: 38.4 North  Longitude: 9 West  Zenith ~= 90  offset: +1 GMT  \*/  echo date("D M d Y"). ', sunrise time : ' .date\_sunrise(time(), SUNFUNCS\_RET\_STRING, 38.4, -9, 90, 1);  ?> |

**|내부함수|**

|  |
| --- |
| mixed **date\_sunset** ( int $timestamp [, int $format = SUNFUNCS\_RET\_STRING [, float $latitude = ini\_get("date.default\_latitude") [, float $longitude = ini\_get("date.default\_longitude") [, float $zenith = ini\_get("date.sunset\_zenith") [, float $gmt\_offset = 0 ]]]]] ) |

내부함수 date\_sunset()는 위치와 날짜에 대한 일몰 시간을 계산합니다.

**예제파일) date\_sunset.php**

|  |
| --- |
| <?php  /\* calculate the sunset time for Lisbon, Portugal  Latitude: 38.4 North  Longitude: 9 West  Zenith ~= 90  offset: +1 GMT  \*/  echo date("D M d Y"). ', sunset time : ' .date\_sunset(time(), SUNFUNCS\_RET\_STRING, 38.4, -9, 90, 1);  ?> |

**05.3 출력포맷**

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

날짜와 시간을 표기하는 방법은 여러 문화권에 따라 다양합니다. PHP는 포맷 표기를 통하여 다양한 형태로 날짜와 시간을 표기할 수 있습니다.

**05.3.1 시간/날짜**

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

내부함수 date() 함수는 지정한 포맷에 맞추어서 시간과 날짜를 표시합니다.

**|내부함수|**

|  |
| --- |
| string **date** ( string $format [, int $timestamp = time() ] ) |

만일 포맷만 입력한 경우에는 현재의 시간을 표시합니다. 두번째 인자로 시간을 표시하면 해당 시간을 포맷 형태로 출력을 합니다.

시간과 날짜를 출력하는 다양한 포맷기호가 있습니다. 시간 포맷을 사용할 때는 대소문자를 구분하여 사용합니다. 각각의 의미와 표시가 다릅니다.

* a: 소문자 am / pm 을 표시합니다.
* A: 대문자 AM / PM 을 표시합니다.
* d: 날짜를 두자리 숫자로 표시합니다. 예) 01, 21 ,31
* D: 요일을 영문약자 3자리로 표시를 합니다. 예) fri, sun, mon
* F: 영문을 달을 표시합니다. 예) January, May, Jun
* m: 달을 숫자 두자리로 표시를 합니다. 예) 01, 03, 12
* M: 다를 영문약자 3자리로 표시합니다. 예) jan, oct, dec
* h: 시간을 12시간 형태로 표시합니다.
* H: 시간을 24시간 형태로 표시합니다.
* i: 분을 표시합니다
* j: 날짜를 표시합니다. 10 이하의 날짜일 경우에는 앞에 0을 표시하지 않습니다.
* l: 영문으로 요일을 나타냅니다. 예) friday, sunday
* s: 초를 두자리로 표시를 합니다.
* S: 영어의 서수를 추가하여 표시하니다. 예) “th”, “nd”
* t: 해당 달의 말일 날짜수를 표시합니다. 예) 28, 30, 31
* U: 특정 기간의후의 초를 표시합니다.
* Y: 연도를 네자리로 표시합니다.
* w: 요일을 숫자로 표시합니다. 예) 일요일=0, 월요일=1, 화요일=2
* y: 연도를 두자리로 표시합니다.
* z: 1년의 날짜 수를 표시합니다.

**예제파일) date.php**

|  |
| --- |
| <?php  // 현재의 날짜와 시간을 표시합니다.  echo date("Y-m-d H:i:s");  echo "<br>";  // 두번째 인자로 주어진 시간을 포맷에 맞게 출력합니다.  echo date("Y-m-d H:i:s",mktime(0,0,0,1,1,2017));  ?> |

**화면출력)**

2017-06-18 06:14:08

2017-01-01 00:00:00

**|내부함수|**

|  |
| --- |
| array **date\_parse\_from\_format** ( string $format , string $date ) |

지정된 형식에 따른 날짜정보 읽어 옵니다.

**예제파일) date\_parse\_from\_format.php**

|  |
| --- |
| <?php  $date = "6.1.2017 13:00+01:00";  print\_r(date\_parse\_from\_format("j.n.Y H:iP", $date));  ?> |

**화면출력)**

Array ( [year] => 2017 [month] => 1 [day] => 6 [hour] => 13 [minute] => 0 [second] => 0 [fraction] => [warning\_count] => 0 [warnings] => Array ( ) [error\_count] => 0 [errors] => Array ( ) [is\_localtime] => 1 [zone\_type] => 1 [zone] => -60 [is\_dst] => )

**05.3.2 포맷**

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

내부함수 strftime() 함수는 현재의 시간과 날짜를 지정한 타입으로 변환하여 출력합니다.

**|내부함수|**

|  |
| --- |
| string **strftime** ( string $format [, int $timestamp = time() ] ) |

**포맷기호:**

* % a 일요일부터 토요일까지의 약식 텍스트 표현
* % A 일요일부터 토요일까지의 전체 텍스트 표현
* % d 두 자리 날짜 (맨 앞에 0이 붙음) 01 – 31
* % e 달의 한 자리이며 한 자리 앞의 공백이 있습니다.
* % j 일, 001에서 366 사이의 0을 시작하는 3 자리 숫자
* % u ISO-8601 요일의 숫자 표현 1 (월요일) ~ 7 (일요일)
* % w 요일의 숫자 표현 0 (일요일 기준) ~ 6 (토요일 기준)

**주간 표시:**

* % U 첫 번째 일요일을 첫 번째 주로 시작하는 해당 연도의 주 번호 13 (해당 연도의 13 번째 주 전체)
* % V ISO-8601 : 주어진 주를 1988 년 주 숫자로, 첫 주부터 시작하여 주 4 일 이상, 월요일은 주 01 ~ 53 주 (여기서 겹치는 주가는 53 일입니다)
* % W 첫 번째 월요일을 첫 번째 주로 시작하는 숫자의 주 표시입니다. 46 (월요일로 시작하는 연도의 46 번째 주)

**날짜표시:**

* % b 1 월에서 12 월까지 로컬을 기반으로 한 약식 월 이름
* % B 1 월에서 12 월까지의 로컬을 기반으로 한 전체의 월 이름
* % h 로케일 (% b의 별칭)을 기준으로 한 약식 월 이름 1 월에서 12 월까지
* % m 이달의 두 자리 표현 01 (1 월분) ~ 12 (12 월분)

**년도 표시:**

* % C 세기의 두 자리 표현
* % g ISO-8601 : 1988 표준 (% V 참조)에 의한 연도의 두 자리 표시
* % G 전체 네 자리 버전의 % g
* % y 연도의 두 자릿수 표현
* % Y 연도에 대한 네 자리 표시

**시간:**

* % H 24 시간제 형식의 시를 00 자리에서 23 자리로 두 자리로 표시합니다.
* % k 24 시간 형식의 시간 (한 자리 앞의 공백은 0에서 23까지)
* % I 12 자리 형식의 두 자리 표시 01 – 12
* % l (소문자 'L') 시간은 12 시간 형식이며 한 자리 숫자는 1에서 12 사이입니다
* % M 분 표시 두자리 수 00 – 59
* % p 대문자 'AM'또는 'PM'(예 : 00:31 AM, 22:23 PM)
* % P 소문자 'am'또는 'pm'(예 : 00:31, 22:23)
* % r "% I : % M : % S % p"와 동일 예 : 09:34:17 오후 21:34:17
* % R "% H : % M"과 같습니다. 예 : 오전 12시 35 분 00:35, 오후 4:44 16:44
* % S 두 번째 00에서 59 사이의 두 자리 표시
* % T "% H : % M : % S"와 동일 예 : 오후 9:34:17의 경우 21:34:17
* % X 날짜가없는 로케일 기반의 선호 시간 표현 예 : 03:59:16 또는 15:59:16
* % z 시간대 오프셋입니다.
* % Z 시간대 약어입니다.

**시간 및 일자:**

* %c 로케일에 따른 선호 날짜 및 시간 스탬프 (예 : 2 월 5 일 화요일 00:45:10 2009 년 2 월 5 일 12:45:10 AM)
* % D "% m / % d / % y"와 동일 예 : 2009 년 2 월 5 일 02/05/09
* % F "% Y- % m- % d"와 동일합니다 (예 : 2009-02-05, 2009 년 2 월 5 일)
* % s Unix Epoch 시간 타임 스탬프 (예 : 1979 년 9 월 10 일 305815200 08:40:00 AM)
* % x 로케일에 기초한 선호 날짜 표현, 시간 없음 (예 : 02/05/09 for February 5, 2009)

**그외 포맷처리:**

* % n 개행 문자 ( "\ n")
* % t 탭 문자 ( "\ t")
* %% 리터럴 퍼센트 문자 ( "%")

**예제파일) strftime.php**

|  |
| --- |
| <?php  echo "축약된 요일이름 = ". strftime("%a") ."<br>";  echo "요일이름 = ". strftime("%A") ."<br>";  echo "날짜(10진수) = ". strftime("%d") ."<br>";  echo "Day of the month, with a space preceding single digits. Not implemented as described on Windows. See below for more information. = ". strftime("%e") ."<br>";  echo "Day of the year, 3 digits with leading zeros = ". strftime("%j") ."<br>";  echo "ISO-8601 numeric representation of the day of the week = ". strftime("%u") ."<br>";  echo "Numeric representation of the day of the week = ". strftime("%w") ."<br>";  echo "Week number of the given year, starting with the first Sunday as the first week = ". strftime("%U") ."<br>";  echo "ISO-8601:1988 week number of the given year, starting with the first week of the year with at least 4 weekdays, with Monday being the start of the week 01 through 53 = ". strftime("%V") ."<br>";  echo "A numeric representation of the week of the year, starting with the first Monday as the first week 46 (for the 46th week of the year beginning with a Monday) = ". strftime("%W") ."<br>";  echo "Abbreviated month name, based on the locale = ". strftime("%b") ."<br>";  echo "Full month name, based on the locale = ". strftime("%B") ."<br>";  echo "Abbreviated month name, based on the locale (an alias of %b) = ". strftime("%h") ."<br>";  echo "Two digit representation of the month = ". strftime("%m") ."<br>";  echo "Two digit representation of the century (year divided by 100, truncated to an integer) = ". strftime("%C") ."<br>";  echo "Two digit representation of the year going by ISO-8601:1988 standards (see %V) = ". strftime("%g") ."<br>";  echo "The full four-digit version of %g = ". strftime("%G") ."<br>";  echo "Two digit representation of the year = ". strftime("%y") ."<br>";  echo "Four digit representation for the year = ". strftime("%Y") ."<br>";  echo "Two digit representation of the hour in 24-hour format = ". strftime("%H") ."<br>";  echo "Hour in 24-hour format, with a space preceding single digits = ". strftime("%k") ."<br>";  echo "Two digit representation of the hour in 12-hour format = ". strftime("%I") ."<br>";  echo "Hour in 12-hour format, with a space preceding single digits = ". strftime("%l") ."<br>";  echo "Two digit representation of the minute = ". strftime("%M") ."<br>";  echo "UPPER-CASE 'AM' or 'PM' based on the given time = ". strftime("%P") ."<br>";  echo "lower-case 'am' or 'pm' based on the given time = ". strftime("%p") ."<br>";    echo "Same as %I:%M:%S %p = ". strftime("%r") ."<br>";  echo "Same as %H:%M = ". strftime("%R") ."<br>";  echo "Two digit representation of the second = ". strftime("%S") ."<br>";  echo "Same as %H:%M:%S = ". strftime("%T") ."<br>";  echo "Preferred time representation based on locale, without the date = ". strftime("%X") ."<br>";  echo "The time zone offset. Not implemented as described on Windows. See below for more information. = ". strftime("%z") ."<br>";  echo "The time zone abbreviation. Not implemented as described on Windows. See below for more information. = ". strftime("%Z") ."<br>";    echo "Preferred date and time stamp based on locale = ". strftime("%c") ."<br>";  echo "Same as %m/%d/%y = ". strftime("%D") ."<br>";  echo "Same as %Y-%m-%d (commonly used in database datestamps) = ". strftime("%F") ."<br>";  echo "Unix Epoch Time timestamp (same as the time() function) = ". strftime("%s") ."<br>";    echo "Preferred date representation based on locale, without the time = ". strftime("%x") ."<br>";  echo "A newline character = ". strftime("%n") ."<br>";  echo "A Tab character = ". strftime("%t") ."<br>";  echo "A literal percentage character = ". strftime("%%") ."<br>";  ?> |

**화면출력)**

축약된 요일이름 = Sun

요일이름 = Sunday

날짜(10진수) = 18

Day of the month, with a space preceding single digits. Not implemented as described on Windows. See below for more information. = 18

Day of the year, 3 digits with leading zeros = 169

ISO-8601 numeric representation of the day of the week = 7

Numeric representation of the day of the week = 0

Week number of the given year, starting with the first Sunday as the first week = 25

ISO-8601:1988 week number of the given year, starting with the first week of the year with at least 4 weekdays, with Monday being the start of the week 01 through 53 = 24

A numeric representation of the week of the year, starting with the first Monday as the first week 46 (for the 46th week of the year beginning with a Monday) = 24

Abbreviated month name, based on the locale = Jun

Full month name, based on the locale = June

Abbreviated month name, based on the locale (an alias of %b) = Jun

Two digit representation of the month = 06

Two digit representation of the century (year divided by 100, truncated to an integer) = 20

Two digit representation of the year going by ISO-8601:1988 standards (see %V) = 17

The full four-digit version of %g = 2017

Two digit representation of the year = 17

Four digit representation for the year = 2017

Two digit representation of the hour in 24-hour format = 07

Hour in 24-hour format, with a space preceding single digits =

Two digit representation of the hour in 12-hour format = 07

Hour in 12-hour format, with a space preceding single digits =

Two digit representation of the minute = 02

UPPER-CASE 'AM' or 'PM' based on the given time =

lower-case 'am' or 'pm' based on the given time = AM

Same as %I:%M:%S %p = 07:02:32 AM

Same as %H:%M = 07:02

Two digit representation of the second = 32

Same as %H:%M:%S = 07:02:32

Preferred time representation based on locale, without the date = 07:02:32

The time zone offset. Not implemented as described on Windows. See below for more information. = +0900

The time zone abbreviation. Not implemented as described on Windows. See below for more information. = ���ѹα� ǥ�ؽ�

Preferred date and time stamp based on locale = Sun Jun 18 07:02:32 2017

Same as %m/%d/%y = 06/18/17

Same as %Y-%m-%d (commonly used in database datestamps) = 2017-06-18

Unix Epoch Time timestamp (same as the time() function) =

Preferred date representation based on locale, without the time = 06/18/17

A newline character =

A Tab character =

A literal percentage character = %

**|내부함수|**

|  |
| --- |
| array **strptime** ( string $date , string $format ) |

내부함수 strftime()으로 생성 된 시간/날짜를 파싱합니다. 참고로strptime()는 윈도우 환경에서는 지원하지 않습니다.

**예제파일) strptime.php**

|  |
| --- |
| <?php  $format = '%d/%m/%Y %H:%M:%S';  $strf = strftime($format);  echo "$strf\n";  print\_r(strptime($strf, $format));  ?> |

**화면출력)**

08/08/2017 15:06:04

Array ( [tm\_sec] => 4 [tm\_min] => 6 [tm\_hour] => 15 [tm\_mday] => 8 [tm\_mon] => 7 [tm\_year] => 117 [tm\_wday] => 2 [tm\_yday] => 219 [unparsed] => )

**05.4 유효성**

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

사용자로 직접 날짜를 입력을 받는 경우 유효성을 체크하여 데이터를 처리하면 보다 안전한 처리를 할 수 있습니다.

**|내부함수|**

|  |
| --- |
| bool **checkdate** ( int $month , int $day , int $year ) |

내부함수 checkdate()는 입력한 날짜정보가 유효한지를 검사할 수 있습니다. 날짜 유효성을 체크하여 결과를 논리값을 반환 합니다.

**예제파일) checkdate.php**

|  |
| --- |
| <?php  $year = "2017";  $month = "06";  $day = "33";  if (checkdate($day,$month,$year)){  echo "$year - $month - $day ";  echo "유효한 날짜 입니다.<br>";  } else {  echo "$year - $month - $day ";  echo "정확하지 않은 날짜입니다.<br>";  }  ?> |

**화면출력)**

2017 - 06 - 33 정확하지 않은 날짜입니다.

**05.5 DateTime 클래스**

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

PHP는 내장함수 이외에 날짜, 시간처리를 보다 다양하게 처리를 할 수 있는 클래스를 지원합니다.

**05.5.1 클래스 정의**

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

클래스의 정의 및 구성은 다음과 같습니다:

|  |
| --- |
| DateTime implements DateTimeInterface {  /\* Constants \*/  const string ATOM = "Y-m-d\TH:i:sP" ;  const string COOKIE = "l, d-M-Y H:i:s T" ;  const string ISO8601 = "Y-m-d\TH:i:sO" ;  const string RFC822 = "D, d M y H:i:s O" ;  const string RFC850 = "l, d-M-y H:i:s T" ;  const string RFC1036 = "D, d M y H:i:s O" ;  const string RFC1123 = "D, d M Y H:i:s O" ;  const string RFC2822 = "D, d M Y H:i:s O" ;  const string RFC3339 = "Y-m-d\TH:i:sP" ;  const string RSS = "D, d M Y H:i:s O" ;  const string W3C = "Y-m-d\TH:i:sP" ;  /\* Methods \*/  public \_\_construct ([ string $time = "now" [, DateTimeZone $timezone = NULL ]] )  public DateTime add ( DateInterval $interval )  public static DateTime createFromFormat ( string $format , string $time [, DateTimeZone $timezone ] )  public static array getLastErrors ( void )  public DateTime modify ( string $modify )  public static DateTime \_\_set\_state ( array $array )  public DateTime setDate ( int $year , int $month , int $day )  public DateTime setISODate ( int $year , int $week [, int $day = 1 ] )  public DateTime setTime ( int $hour , int $minute [, int $second = 0 ] )  public DateTime setTimestamp ( int $unixtimestamp )  public DateTime setTimezone ( DateTimeZone $timezone )  public DateTime sub ( DateInterval $interval )  public DateInterval diff ( DateTimeInterface $datetime2 [, bool $absolute = false ] )  public string format ( string $format )  public int getOffset ( void )  public int getTimestamp ( void )  public DateTimeZone getTimezone ( void )  public \_\_wakeup ( void )  } |

**05.5.2 매서드**

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

매서드 diff()는 두 DateTime 객체 간의 차이점을 반환합니다. date\_diff() 함수는 객체지향 클래스 DateTimeZone::diff 의 별칭입니다.

**|매서드|**

|  |
| --- |
| public DateInterval DateTime::**diff** ( DateTimeInterface $datetime2 [, bool $absolute = false ] ) |

**예제파일) date\_diff.php**

|  |
| --- |
| <?php  // Object oriented style  $datetime1 = new DateTime('2017-08-08');  $datetime2 = new DateTime('2017-08-13');  $interval = $datetime1->diff($datetime2);  echo $interval->format('%R%a days');  echo "<br>";  // Procedural style  $datetime1 = date\_create('2017-08-11');  $datetime2 = date\_create('2017-08-13');  $interval = date\_diff($datetime1, $datetime2);  echo $interval->format('%R%a days');  ?> |

**화면출력)**

+5 days

+2 days

**|매서드|**

|  |
| --- |
| public DateTime DateTime::**sub** ( DateInterval $interval ) |

매서드 sub()는 DateTime 객체에서 일, 월, 년,시, 분, 초를 뺍니다. date\_sub() 함수는 객체지향 클래스 DateTimeZone::sub 의 별칭입니다.

**예제파일) date\_sub.php**

|  |
| --- |
| <?php  // Object oriented style  $date = new DateTime('2017-08-20');  $date->sub(new DateInterval('P10D'));  echo $date->format('Y-m-d') . "<br>";  // Procedural style  $date = date\_create('2017-08-20');  date\_sub($date, date\_interval\_create\_from\_date\_string('15 days'));  echo date\_format($date, 'Y-m-d');  ?> |

**화면출력)**

2017-08-10

2017-08-05

**|매서드|**

|  |
| --- |
| public string DateTime::**format** ( string $format ) |

매서드 format()은 주어진 형식에 따라 날짜 형식을 반환합니다. date\_format() 함수는 객체지향 클래스 DateTimeZone::format 의 별칭입니다.

**예제파일) date\_format.php**

|  |
| --- |
| <?php  // Object oriented style  $date = new DateTime('2018-07-01');  echo $date->format('Y-m-d H:i:s');  echo "<br>";  // Procedural style  $date = date\_create('2018-08-01');  echo date\_format($date, 'Y-m-d H:i:s');  ?> |

**화면출력)**

2018-07-01 00:00:00

2018-08-01 00:00:00

**|매서드|**

|  |
| --- |
| public DateTime DateTime::**setISODate** ( int $year , int $week [, int $day = 1 ] ) |

매서드 setISODate()는 ISO 날짜를 설정합니다. date\_isodate\_set() 함수는 객체지향 클래스 DateTimeZone::setISODate 의 별칭입니다.

**예제파일) date\_isodate\_set.php**

|  |
| --- |
| <?php  // Object oriented style  $date = new DateTime();  $date->setISODate(2017, 2);  echo $date->format('Y-m-d') . "<br>";  $date->setISODate(2017, 2, 7);  echo $date->format('Y-m-d') . "<br>";  // Procedural style  $date = date\_create();  date\_isodate\_set($date, 2018, 2);  echo date\_format($date, 'Y-m-d') . "<br>";  date\_isodate\_set($date, 2018, 2, 7);  echo date\_format($date, 'Y-m-d') . "<br>";  ?> |

**화면출력)**

2017-01-09

2017-01-15

2018-01-08

2018-01-14

**|매서드|**

|  |
| --- |
| public DateTime DateTime::**setTime** ( int $hour , int $minute [, int $second = 0 ] ) |

매서드 setTime()는 시간을 설정합니다. date\_time\_set() 함수는 객체지향 클래스 DateTimeZone::setTime 의 별칭입니다.

**예제파일) date\_time\_set.php**

|  |
| --- |
| <?php  // Object oriented style  $date = new DateTime('2017-08-08');  $date->setTime(14, 55);  echo $date->format('Y-m-d H:i:s') . "<br>";  $date->setTime(14, 55, 24);  echo $date->format('Y-m-d H:i:s') . "<br>";  // Procedural style  $date = date\_create('2017-08-09');  date\_time\_set($date, 14, 55);  echo date\_format($date, 'Y-m-d H:i:s') . "<br>";  date\_time\_set($date, 14, 55, 24);  echo date\_format($date, 'Y-m-d H:i:s') . "<br>";  ?> |

**화면출력)**

2017-08-08 14:55:00

2017-08-08 14:55:24

2017-08-09 14:55:00

2017-08-09 14:55:24

**|매서드|**

|  |
| --- |
| public DateTime DateTime::modify ( string $modify ) |

매서드 modify()는 타임 스탬프를 변경합니다. date\_modify() 함수는 객체지향 클래스 DateTimeZone::modify 의 별칭입니다.

**예제파일) date\_modify.php**

|  |
| --- |
| <?php  // Object oriented style  $date = new DateTime('2017-12-12');  $date->modify('+1 day');  echo $date->format('Y-m-d');  echo "<br>";  // Procedural style  $date = date\_create('2018-12-24');  date\_modify($date, '+1 day');  echo date\_format($date, 'Y-m-d');  ?> |

**화면출력)**

2017-12-13

2018-12-25

**|매서드|**

|  |
| --- |
| public int DateTime::**getTimestamp** ( void ) |

매서드 getTimestamp()는 Unix 타임 스탬프를 가져옵니다. date\_timestamp\_get() 함수는 객체지향 클래스 DateTimeZone::getTimestamp 의 별칭입니다.

**예제파일) date\_timestamp\_get.php**

|  |
| --- |
| <?php  // Object oriented style  $date = new DateTime();  echo $date->getTimestamp();  echo "<br>";    // Procedural style  $date = date\_create();  echo date\_timestamp\_get($date);  ?> |

**화면출력)**

1502176161

1502176161

**|매서드|**

|  |
| --- |
| public DateTime DateTime::**setTimestamp** ( int $unixtimestamp ) |

매서드 setTimestamp()는 Unix 타임 스탬프를 으로 날짜와 시간을 설정합니다. date\_timestamp\_set() 함수는 객체지향 클래스 DateTimeZone::setTimestamp 의 별칭입니다.

**예제파일) date\_timestamp\_set.php**

|  |
| --- |
| <?php  // Object oriented style  $date = new DateTime();  echo $date->format('U = Y-m-d H:i:s') . "<br>";  $date->setTimestamp(1502176161);  echo $date->format('U = Y-m-d H:i:s') . "<br>";  // Procedural style  $date = date\_create();  echo date\_format($date, 'U = Y-m-d H:i:s') . "<br>";  date\_timestamp\_set($date, 1502176161);  echo date\_format($date, 'U = Y-m-d H:i:s') . "<br>";  ?> |

**화면출력)**

1502176395 = 2017-08-08 07:13:15

1502176161 = 2017-08-08 07:09:21

1502176395 = 2017-08-08 07:13:15

1502176161 = 2017-08-08 07:09:21

**|매서드|**

|  |
| --- |
| public DateTime DateTime::**setTimezone** ( DateTimeZone $timezone ) |

매서드 setTimezone()는 DateTime 표준 시간대를 설정합니다. date\_timezone\_set() 함수는 객체지향 클래스 DateTimeZone::setTimezone 의 별칭입니다.

**예제파일) date\_timezone\_set.php**

|  |
| --- |
| <?php  // Object oriented style  $date = new DateTime('2017-08-01', new DateTimeZone('Asia/Seoul'));  echo $date->format('Y-m-d H:i:sP') . "<br>";  $date->setTimezone(new DateTimeZone('Europe/London'));  echo $date->format('Y-m-d H:i:sP') . "<br>";  // Procedural style  $date = date\_create('2017-08-08', timezone\_open('Asia/Tokyo'));  echo date\_format($date, 'Y-m-d H:i:sP') . "<br>";  date\_timezone\_set($date, timezone\_open('America/New\_York'));  echo date\_format($date, 'Y-m-d H:i:sP') . "<br>";  ?> |

**화면출력)**

2017-08-01 00:00:00+09:00

2017-07-31 16:00:00+01:00

2017-08-08 00:00:00+09:00

2017-08-07 11:00:00-04:00

**|매서드|**

|  |
| --- |
| public DateTimeZone DateTime::**getTimezone** ( void ) |

매서드 getTimezone()는 DateTime을 기준으로 시간대를 반환합니다. date\_timezone\_get() 함수는 객체지향 클래스 DateTimeZone::getTimezone 의 별칭입니다.

**예제파일) date\_timezone\_get.php**

|  |
| --- |
| <?php  // Object oriented style  $date = new DateTime(null, new DateTimeZone('Asia/Seoul'));  $tz = $date->getTimezone();  echo $tz->getName();  echo "<br>";    // Procedural style  $date = date\_create(null, timezone\_open('America/New\_York'));  $tz = date\_timezone\_get($date);  echo timezone\_name\_get($tz);  ?> |

**화면출력)**

Asia/Seoul

America/New\_York

**|매서드|**

|  |
| --- |
| public int DateTime::**getOffset** ( void ) |

매서드 getOffset()는 시간대 오프셋을 반환합니다. date\_offset\_get() 함수는 객체지향 클래스 DateTimeZone::getOffset 의 별칭입니다.

**예제파일) date\_offset\_get.php**

|  |
| --- |
| <?php  // Object oriented style  $winter = new DateTime('2018-12-21', new DateTimeZone('America/New\_York'));  $summer = new DateTime('2017-08-8', new DateTimeZone('America/New\_York'));  echo $winter->getOffset() . "<br>";  echo $summer->getOffset() . "<br>";  // Procedural style  $winter = date\_create('2018-12-21', timezone\_open('America/New\_York'));  $summer = date\_create('2017-08-9', timezone\_open('America/New\_York'));  echo date\_offset\_get($winter) . "<br>";  echo date\_offset\_get($summer) . "<br>";  ?> |

**화면출력)**

-18000

-14400

-18000

-14400

**|매서드|**

|  |
| --- |
| public static array DateTime::**getLastErrors** ( void ) |

매서드 getLastErrors()는 경고 및 오류를 반환합니다. date\_get\_last\_errors() 함수는 객체지향 클래스 DateTimeZone::getLastErrorst 의 별칭입니다.

**예제파일) date\_get\_last\_errors.php**

|  |
| --- |
| <?php  // Object oriented style  try {  $date = new DateTime('asdfasdf');  } catch (Exception $e) {  // 데모 용..  print\_r(DateTime::getLastErrors());  // 실제 객체 지향적 인 방법  // echo $e->getMessage();  }    //Procedural style  $date = date\_create('asdfasdf');  print\_r(date\_get\_last\_errors());  ?> |

**화면출력)**

Array ( [warning\_count] => 1 [warnings] => Array ( [6] => Double timezone specification ) [error\_count] => 1 [errors] => Array ( [0] => The timezone could not be found in the database ) ) Array ( [warning\_count] => 1 [warnings] => Array ( [6] => Double timezone specification ) [error\_count] => 1 [errors] => Array ( [0] => The timezone could not be found in the database ) )

**05.6 달력**

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

PHP의 달력 확장기능은 달력처리를 쉽고 간소하게 하기 위한 함수들을 지원합니다. 달력함수들은 BC 4713 부터 줄리안 데이 카운트 (Julian Day Count)를 기준으로합니다.

달력 형식을 변환하려면 먼저 줄리안 일 수로 변환 한 다음 원하는 달력으로 변환해야합니다. Julian 일수와 Julian 달력은 같지 않습니다.

**05.6.1 설정**

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

본 기능을 사용하기 위해서는 --enable-calendar 옵션을 적용하여 컴파일 되어야 합니다. 윈도우버젼은 확장기능으로 빌드인 되어 있습니다.

**|내부함수|**

|  |
| --- |
| int **cal\_days\_in\_month** ( int $calendar , int $month , int $year ) |

내부함수 cal\_days\_in\_month()는 지정한 년도와 달의 마지막 일수를 반환합니다.

**예제파일) cal\_days\_in\_month.php**

|  |
| --- |
| <?php  $number = cal\_days\_in\_month(CAL\_GREGORIAN, 8, 2017);  echo "2017년 8월의 마지막 일자는 {$number} 입니다.";  ?> |

**화면출력)**

2017년 8월의 마지막 일자는 31 입니다.

**|내부함수|**

|  |
| --- |
| array **cal\_from\_jd** ( int $jd , int $calendar ) |

내부함수 cal\_from\_jd()는 Julian Day Count를 지정된 달력의 날짜로 변환합니다.

**예제파일) cal\_from\_jd.php**

|  |
| --- |
| <?php  $today = unixtojd(mktime(0, 0, 0, 8, 6, 2017));  print\_r(cal\_from\_jd($today, CAL\_GREGORIAN));  ?> |

**화면출력)**

Array ( [date] => 8/6/2017 [month] => 8 [day] => 6 [year] => 2017 [dow] => 0 [abbrevdayname] => Sun [dayname] => Sunday [abbrevmonth] => Aug [monthname] => August )

**|내부함수|**

|  |
| --- |
| array **cal\_info** ([ int $calendar = -1 ] ) |

내부함수 cal\_info()는 캘린더에 대한 정보를 반환합니다.

* 0 또는 CAL\_GREGORIAN - Gregorian Calendar
* 1 또는 CAL\_JULIAN - Julian Calendar
* 2 또는 CAL\_JEWISH - Jewish Calendar
* 3 또는 CAL\_FRENCH - French Revolutionary Calendar

**예제파일) cal\_info.php**

|  |
| --- |
| <?php  $info = cal\_info(0);  print\_r($info);  ?> |

**화면출력)**

Array (

[months] => Array ( [1] => January [2] => February [3] => March [4] => April [5] => May [6] => June [7] => July [8] => August [9] => September [10] => October [11] => November [12] => December )

[abbrevmonths] => Array ( [1] => Jan [2] => Feb [3] => Mar [4] => Apr [5] => May [6] => Jun [7] => Jul [8] => Aug [9] => Sep [10] => Oct [11] => Nov [12] => Dec )

[maxdaysinmonth] => 31

[calname] => **Gregorian**

[calsymbol] => CAL\_GREGORIAN )

**05.6.2 Julian**

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

내부함수 cal\_to\_jd()는 달력에서 줄리안 데이 수를 반환합니다.

**|내부함수|**

|  |
| --- |
| int **cal\_to\_jd** ( int $calendar , int $month , int $day , int $year ) |

**예제파일) cal\_to\_jd.php**

|  |
| --- |
| <?php  echo cal\_to\_jd( CAL\_GREGORIAN , 8 , 6 , 2017 );  ?> |

**화면출력)**

2457972

**|내부함수|**

|  |
| --- |
| int **gregoriantojd** ( int $month , int $day , int $year ) |

내부함수 gregoriantojd()는 Gregorian Calendar 기준 Julian Day 수를 반환합니다.

**예제파일) gregoriantojd.php**

|  |
| --- |
| <?php  echo gregoriantojd( 8 , 6 , 2017 );  ?> |

**화면출력)**

2457972

**|내부함수|**

|  |
| --- |
| mixed **jddayofweek** ( int $julianday [, int $mode = CAL\_DOW\_DAYNO ] ) |

내부함수 jddayofweek()는 요일을 반환합니다.

**예제파일) jddayofweek.php**

|  |
| --- |
| <?php  $julianday = gregoriantojd( 8 , 6 , 2017 );  // Return the day number as an int (0=Sunday, 1=Monday, etc)  echo jddayofweek($julianday) ."<br>";  // 1 Returns string containing the day of week (English-Gregorian)  echo jddayofweek($julianday,1) ."<br>";  // 2 Return a string containing the abbreviated day of week (English-Gregorian)  echo jddayofweek($julianday,2) ."<br>";  ?> |

**화면출력)**

0

Sunday

Sun

**|내부함수|**

|  |
| --- |
| string **jdmonthname** ( int $julianday , int $mode ) |

내부함수 jdmonthname()는 월 이름을 반환합니다.

**예제파일) jdmonthname.php**

|  |
| --- |
| <?php  $julianday = gregoriantojd( 8 , 6 , 2017 );  // 0  // Gregorian - abbreviated Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec  echo jdmonthname ( $julianday , 0 ) ."<br>";    // 1  // Gregorian January, February, March, April, May, June, July, August, September, October, November, December  echo jdmonthname ( $julianday , 1 ) ."<br>";    // 2  // Julian - abbreviated Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec  echo jdmonthname ( $julianday , 2 ) ."<br>";    // 3  // Julian January, February, March, April, May, June, July, August, September, October, November, December  echo jdmonthname ( $julianday , 3 ) ."<br>";    // 4  // Jewish Tishri, Heshvan, Kislev, Tevet, Shevat, AdarI, AdarII, Nisan, Iyyar, Sivan, Tammuz, Av, Elul  echo jdmonthname ( $julianday , 4 ) ."<br>";  // 5  // French Republican Vendemiaire, Brumaire, Frimaire, Nivose, Pluviose, Ventose, Germinal, Floreal, Prairial, Messidor, Thermidor, Fructidor, Extra  echo jdmonthname ( $julianday , 5 ) ."<br>";  ?> |

**화면출력)**

Aug

August

Jul

July

Av

**|내부함수|**

|  |
| --- |
| string **jdtojulian** ( int $julianday ) |

내부함수 jdtojulian()는 Julian 일수를 Julian 날짜로 변환 합니다.

**|내부함수|**

|  |
| --- |
| int **juliantojd** ( int $month , int $day , int $year ) |

내부함수 juliantojd()는 Julian 날짜를 Julian 일수로 변환합니다.

**|내부함수|**

|  |
| --- |
| int **jdtounix** ( int $jday ) |

내부함수 jdtounix()는 Julian 일수를 Unix 타임 스탬프로 변환합니다.

**|내부함수|**

|  |
| --- |
| int **unixtojd** ([ int $timestamp = time() ] ) |

내부함수 unixtojd()는 Unix 타임 스탬프를 Julian 일 수로 변환합니다.

**05.6.3 Gregorian**

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

Gregorian 기준의 날짜를 변환하여 사용할 수 있습니다. 이와 관련된 몇가지 함수들을 지원합니다.

**|내부함수|**

|  |
| --- |
| string **jdtogregorian** ( int $julianday ) |

내부함수 jdtogregorian()는 Julian 일 수를 Gregorian 날짜로 변환 합니다.

**|내부함수|**

|  |
| --- |
| int **gregoriantojd** ( int $month , int $day , int $year ) |

내부함수 gregoriantojd()는 Gregorian 날짜를 Julian 일 수로 변환 합니다.

**|내부함수|**

|  |
| --- |
| string **jdtogregorian** ( int $julianday ) |

내부함수 jdtogregorian()는 Julian 일수를 Gregorian 날짜로 변환 합니다.

**05.6.4 Jewish**

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

Jewish 기준의 날짜를 변환하여 사용할 수 있습니다. 이와 관련된 몇가지 함수들을 지원합니다.

**|내부함수|**

|  |
| --- |
| string **jdtojewish** ( int $juliandaycount [, bool $hebrew = false [, int $fl = 0 ]] ) |

내부함수 jdtojewish()는 Julian 일 수를 Jewish 달력일자로 변환 합니다.

**|내부함수|**

|  |
| --- |
| int **jewishtojd** ( int $month , int $day , int $year ) |

내부함수 jewishtojd()는 Jewish 날짜를 Julian 일수로 변환 합니다.

**05.6.4 French Republican**

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

French Republican 기준의 날짜를 변환하여 사용할 수 있습니다. 이와 관련된 몇가지 함수들을 지원합니다.

**|내부함수|**

|  |
| --- |
| string **jdtofrench** ( int $juliandaycount ) |

French Republican Calendar 를 Julian 일수로 변환 합니다.

**예제파일) jdtofrench.php**

|  |
| --- |
| <?php  $julianday = gregoriantojd( 8 , 6 , 2017 );  echo jdtofrench($julianday);  ?> |

**|내부함수|**

|  |
| --- |
| int **frenchtojd** ( int $month , int $day , int $year ) |

내부함수 frenchtojd()는 French Revolutionary Calendar 기준 Julian Day 수를 반환합니다.

**예제파일) frenchtojd.php**

|  |
| --- |
| <?php  echo frenchtojd( 8 , 6 , 2017 );  ?> |

**05.6.6 부활절**

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

부활절 관련 함수들을 지원합니다.

**|내부함수|**

|  |
| --- |
| int **easter\_date** ([ int $year = date("Y") ] ) |

내부함수 easter\_date()는 지정한 년도의 부활절 자정에 대한 유닉스 타임스탬프를 반환합니다.

**예제파일) easter\_date.php**

|  |
| --- |
| <?php  echo "2000 부활절 자정 = ". date("M-d-Y", easter\_date(2000));  echo "<br>";    echo "2010 부활절 자정 = ". date("M-d-Y", easter\_date(2010));  echo "<br>";    echo "2017 부활절 자정 = ". date("M-d-Y", easter\_date(2017));  echo "<br>";  ?> |

**화면출력)**

2000 부활절 자정 = Apr-22-2000

2010 부활절 자정 = Apr-03-2010

2017 부활절 자정 = Apr-15-2017

**|내부함수|**

|  |
| --- |
| int **easter\_days** ([ int $year = date("Y") [, int $method = CAL\_EASTER\_DEFAULT ]] ) |

내부함수 easter\_days()는 부활절의 해에 3 월 21 일 이후 일 수를 반환합니다.

**예제파일) easter\_days.php**

|  |
| --- |
| <?php  echo easter\_days(1999) ."<br>";  // 14, i.e. April 4    echo easter\_days(1492) ."<br>";  // 32, i.e. April 22  echo easter\_days(1913) ."<br>";  // 2, i.e. March 23  ?> |

**화면출력)**

14

32

2

**05.7 타임존**

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

서비스를 전세계 대상으로 대상으로 기획을 할다고 하면, 지역별 시간대 관리는 매우 중요합니다. 각 대륙, 지역별로 구분하여 시간관리 프로그램을 만들어야 정확한 월드와이드 서비스를 구축할 수 있습니다.

**05.7.1 클래스**

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

PHP는 타임존을 관리 및 설정할 수 있는 DateTimeZone 클래스를 지원합니다.

|  |
| --- |
| DateTimeZone {  /\* Constants \*/  const integer AFRICA = 1 ;  const integer AMERICA = 2 ;  const integer ANTARCTICA = 4 ;  const integer ARCTIC = 8 ;  const integer ASIA = 16 ;  const integer ATLANTIC = 32 ;  const integer AUSTRALIA = 64 ;  const integer EUROPE = 128 ;  const integer INDIAN = 256 ;  const integer PACIFIC = 512 ;  const integer UTC = 1024 ;  const integer ALL = 2047 ;  const integer ALL\_WITH\_BC = 4095 ;  const integer PER\_COUNTRY = 4096 ;  /\* Methods \*/  public \_\_construct ( string $timezone )  public array getLocation ( void )  public string getName ( void )  public int getOffset ( DateTime $datetime )  public array getTransitions ([ int $timestamp\_begin [, int $timestamp\_end ]] )  public static array listAbbreviations ( void )  public static array listIdentifiers ([ int $what = DateTimeZone::ALL [, string $country = NULL ]] )  } |

**05.7.2 메서드**

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

**|매서드|**

|  |
| --- |
| public DateTimeZone::**\_\_construct** ( string $timezone ) |

매서드 \_\_construct()는 새 DateTimeZone 객체를 만듭니다. timezone\_open() 함수는 객체지향 클래스 DateTimeZone::\_\_construct 의 별칭입니다.

**예제파일) timezone\_open.php**

|  |
| --- |
| <?php  // Error handling by catching exceptions  $timezones = array('Europe/London', 'Mars/Phobos', 'Jupiter/Europa');  foreach ($timezones as $tz) {  try {  $mars = new DateTimeZone($tz);  } catch(Exception $e) {  echo $e->getMessage() . '<br />';  }  }  ?> |

**화면출력)**

DateTimeZone::\_\_construct(): Unknown or bad timezone (Mars/Phobos)

DateTimeZone::\_\_construct(): Unknown or bad timezone (Jupiter/Europa)

**|매서드|**

|  |
| --- |
| public array DateTimeZone::**getLocation** ( void ) |

매서드 getLocation()는 시간대의 위치 정보를 반환합니다. timezone\_location\_get() 함수는 객체지향 클래스 DateTimeZone::getLocation 의 별칭입니다. 국가 코드, 위도 / 경도 및 주석을 포함하여 시간대의 위치 정보를 반환합니다.

**예제파일) timezone\_location\_get.php**

|  |
| --- |
| <?php  $tz = new DateTimeZone("Europe/Prague");  print\_r($tz->getLocation());  print\_r(timezone\_location\_get($tz));  ?> |

**화면출력)**

Array ( [country\_code] => CZ [latitude] => 50.08333 [longitude] => 14.43333 [comments] => ) Array ( [country\_code] => CZ [latitude] => 50.08333 [longitude] => 14.43333 [comments] => )

**|매서드|**

|  |
| --- |
| public string DateTimeZone::**getName** ( void ) |

매서드 getName()는 타임 존의 이름을 반환합니다. timezone\_name\_get() 함수는 객체지향 클래스 DateTimeZone::getName 의 별칭입니다.

**|매서드|**

|  |
| --- |
| public static array DateTimeZone::**listAbbreviations** ( void ) |

매서드 listAbbreviations()는 dst, offset 및 시간대 이름이 들어있는 연관 배열을 반환합니다. timezone\_abbreviations\_list() 함수는 객체지향 클래스 DateTimeZone::listAbbreviations 의 별칭입니다.

**예제파일) timezone\_abbreviations\_list.php**

|  |
| --- |
| <?php  $timezone\_abbreviations = DateTimeZone::listAbbreviations();  print\_r($timezone\_abbreviations["acst"]);  ?> |

**화면출력)**

Array ( [0] => Array ( [dst] => 1 [offset] => -14400 [timezone\_id] => America/Porto\_Acre ) [1] => Array ( [dst] => [offset] => 32400 [timezone\_id] => Australia/Adelaide ) [2] => Array ( [dst] => [offset] => 34200 [timezone\_id] => Australia/Adelaide ) [3] => Array ( [dst] => 1 [offset] => -14400 [timezone\_id] => America/Eirunepe ) [4] => Array ( [dst] => 1 [offset] => -14400 [timezone\_id] => America/Rio\_Branco ) [5] => Array ( [dst] => 1 [offset] => -14400 [timezone\_id] => Brazil/Acre ) [6] => Array ( [dst] => [offset] => 32400 [timezone\_id] => Australia/Broken\_Hill ) [7] => Array ( [dst] => [offset] => 32400 [timezone\_id] => Australia/Darwin ) [8] => Array ( [dst] => [offset] => 32400 [timezone\_id] => Australia/North ) [9] => Array ( [dst] => [offset] => 32400 [timezone\_id] => Australia/South ) [10] => Array ( [dst] => [offset] => 32400 [timezone\_id] => Australia/Yancowinna ) [11] => Array ( [dst] => [offset] => 34200 [timezone\_id] => Asia/Jayapura ) [12] => Array ( [dst] => [offset] => 34200 [timezone\_id] => Australia/Broken\_Hill ) [13] => Array ( [dst] => [offset] => 34200 [timezone\_id] => Australia/Darwin ) [14] => Array ( [dst] => [offset] => 34200 [timezone\_id] => Australia/North ) [15] => Array ( [dst] => [offset] => 34200 [timezone\_id] => Australia/South ) [16] => Array ( [dst] => [offset] => 34200 [timezone\_id] => Australia/Yancowinna ) )

**|매서드|**

|  |
| --- |
| public static array DateTimeZone::**listIdentifiers** ([ int $what = DateTimeZone::ALL [, string $country = NULL ]] ) |

매서드 listIdentifiers()는 정의 된 모든 시간대 식별자가 포함 된 숫자로 색인 된 배열을 반환합니다. timezone\_identifiers\_list() 함수는 객체지향 클래스 DateTimeZone::listIdentifiers 의 별칭입니다.

**예제파일) timezone\_identifiers\_list.php**

|  |
| --- |
| <?php  $timezone\_identifiers = DateTimeZone::listIdentifiers();  for ($i=0; $i < 5; $i++) {  echo "$timezone\_identifiers[$i]<br>";  }  ?> |

**화면출력)**

Africa/Abidjan

Africa/Accra

Africa/Addis\_Ababa

Africa/Algiers

Africa/Asmara

**|매서드|**

|  |
| --- |
| public int DateTimeZone::**getOffset** ( DateTime $datetime ) |

매서드 getOffset()는 GMT로부터의 시간대 오프셋을 반환합니다. timezone\_offset\_get() 함수는 객체지향 클래스 DateTimeZone::getOffset 의 별칭입니다.

**예제파일) timezone\_offset\_get.php**

|  |
| --- |
| <?php  $dateTimeZoneAmsterdam= new DateTimeZone("Europe/Amsterdam");  $dateTimeZoneSeoul = new DateTimeZone("Asia/Seoul");  $dateTimeAmsterdam = new DateTime("now", $dateTimeZoneAmsterdam);  $dateTimeSeoul = new DateTime("now", $dateTimeZoneSeoul);  $timeOffset = $dateTimeZoneSeoul->getOffset($dateTimeAmsterdam);  var\_dump($timeOffset);  ?> |

**화면출력)**

int(32400)

**|매서드|**

|  |
| --- |
| public array DateTimeZone::**getTransitions** ([ int $timestamp\_begin [, int $timestamp\_end ]] ) |

매서드 getTransitions()는 시간대의 모든 전환을 반환합니다. timezone\_transitions\_get() 함수는 객체지향 클래스 DateTimeZone::getTransitions 의 별칭입니다.

**예제파일) timezone\_transitions\_get.php**

|  |
| --- |
| <?php  $timezone = new DateTimeZone("Europe/London");  $transitions = $timezone->getTransitions();  print\_r(array\_slice($transitions, 0, 3));  ?> |

**화면출력)**

Array ( [0] => Array ( [ts] => -2147483648 [time] => 1901-12-13T20:45:52+0000 [offset] => -75 [isdst] => [abbr] => LMT ) [1] => Array ( [ts] => -2147483648 [time] => 1901-12-13T20:45:52+0000 [offset] => 0 [isdst] => [abbr] => GMT ) [2] => Array ( [ts] => -1691964000 [time] => 1916-05-21T02:00:00+0000 [offset] => 3600 [isdst] => 1 [abbr] => BST ) )

**|내부함수|**

|  |
| --- |
| string **timezone\_version\_get** ( void ) |

내부함수 timezone\_version\_get()는 timezonedb의 버전을 가져옵니다.

**예제파일) timezone\_version\_get.php**

|  |
| --- |
| <?php  echo timezone\_version\_get();  ?> |

**화면출력)**

2017.2

**|내부함수|**

|  |
| --- |
| bool **date\_default\_timezone\_set** ( string $timezone\_identifier ) |

내부함수 date\_default\_timezone\_set()는 날짜 / 시간 함수에서 사용하는 기본 시간대를 설정합니다.

**|내부함수|**

|  |
| --- |
| string **date\_default\_timezone\_get** ( void ) |

내부함수 date\_default\_timezone\_get()는 날짜 / 시간 함수에서 사용하는 기본 표준 시간대를 가져옵니다.

**예제파일) date\_default\_timezone\_get.php**

|  |
| --- |
| <?php  date\_default\_timezone\_set('Asia/Seoul');  if (**date\_default\_timezone\_get()**) {  echo 'date\_default\_timezone\_set: ' . date\_default\_timezone\_get() . '<br />';  }  if (ini\_get('date.timezone')) {  echo 'date.timezone: ' . ini\_get('date.timezone');  }  ?> |

**화면출력)**

date\_default\_timezone\_set: Asia/Seoul

**05.7.3 타임존 명칭**

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

전세계 타이존은 다음과 같이 구분할 수 있습니다.

**Africa**

Africa/Abidjan Africa/Accra Africa/Addis\_Ababa Africa/Algiers Africa/Asmara

Africa/Asmera Africa/Bamako Africa/Bangui Africa/Banjul Africa/Bissau

Africa/Blantyre Africa/Brazzaville Africa/Bujumbura Africa/Cairo Africa/Casablanca

Africa/Ceuta Africa/Conakry Africa/Dakar Africa/Dar\_es\_Salaam Africa/Djibouti

Africa/Douala Africa/El\_Aaiun Africa/Freetown Africa/Gaborone Africa/Harare

Africa/Johannesburg Africa/Juba Africa/Kampala Africa/Khartoum Africa/Kigali

Africa/Kinshasa Africa/Lagos Africa/Libreville Africa/Lome Africa/Luanda

Africa/Lubumbashi Africa/Lusaka Africa/Malabo Africa/Maputo Africa/Maseru

Africa/Mbabane Africa/Mogadishu Africa/Monrovia Africa/Nairobi Africa/Ndjamena

Africa/Niamey Africa/Nouakchott Africa/Ouagadougou Africa/Porto-Novo Africa/Sao\_Tome

Africa/Timbuktu Africa/Tripoli Africa/Tunis Africa/Windhoek

**America**

America/Adak America/Anchorage America/Anguilla

America/Antigua America/Araguaina America/Argentina/Buenos\_Aires

America/Argentina/Catamarca America/Argentina/ComodRivadavia America/Argentina/Cordoba

America/Argentina/Jujuy America/Argentina/La\_Rioja America/Argentina/Mendoza

America/Argentina/Rio\_Gallegos America/Argentina/Salta America/Argentina/San\_Juan

America/Argentina/San\_Luis America/Argentina/Tucuman America/Argentina/Ushuaia

America/Aruba America/Asuncion America/Atikokan

America/Atka America/Bahia America/Bahia\_Banderas

America/Barbados America/Belem America/Belize

America/Blanc-Sablon America/Boa\_Vista America/Bogota

America/Boise America/Buenos\_Aires America/Cambridge\_Bay

America/Campo\_Grande America/Cancun America/Caracas

America/Catamarca America/Cayenne America/Cayman

America/Chicago America/Chihuahua America/Coral\_Harbour

America/Cordoba America/Costa\_Rica America/Creston

America/Cuiaba America/Curacao America/Danmarkshavn

America/Dawson America/Dawson\_Creek America/Denver

America/Detroit America/Dominica America/Edmonton

America/Eirunepe America/El\_Salvador America/Ensenada

America/Fort\_Wayne America/Fortaleza America/Glace\_Bay

America/Godthab America/Goose\_Bay America/Grand\_Turk

America/Grenada America/Guadeloupe America/Guatemala

America/Guayaquil America/Guyana America/Halifax

America/Havana America/Hermosillo America/Indiana/Indianapolis

America/Indiana/Knox America/Indiana/Marengo America/Indiana/Petersburg

America/Indiana/Tell\_City America/Indiana/Vevay America/Indiana/Vincennes

America/Indiana/Winamac America/Indianapolis America/Inuvik

America/Iqaluit America/Jamaica America/Jujuy

America/Juneau America/Kentucky/Louisville America/Kentucky/Monticello

America/Knox\_IN America/Kralendijk America/La\_Paz

America/Lima America/Los\_Angeles America/Louisville

America/Lower\_Princes America/Maceio America/Managua

America/Manaus America/Marigot America/Martinique

America/Matamoros America/Mazatlan America/Mendoza

America/Menominee America/Merida America/Metlakatla

America/Mexico\_City America/Miquelon America/Moncton

America/Monterrey America/Montevideo America/Montreal

America/Montserrat America/Nassau America/New\_York

America/Nipigon America/Nome America/Noronha

America/North\_Dakota/Beulah America/North\_Dakota/Center America/North\_Dakota/New\_Salem

America/Ojinaga America/Panama America/Pangnirtung

America/Paramaribo America/Phoenix America/Port-au-Prince

America/Port\_of\_Spain America/Porto\_Acre America/Porto\_Velho

America/Puerto\_Rico America/Rainy\_River America/Rankin\_Inlet

America/Recife America/Regina America/Resolute

America/Rio\_Branco America/Rosario America/Santa\_Isabel

America/Santarem America/Santiago America/Santo\_Domingo

America/Sao\_Paulo America/Scoresbysund America/Shiprock

America/Sitka America/St\_Barthelemy America/St\_Johns

America/St\_Kitts America/St\_Lucia America/St\_Thomas

America/St\_Vincent America/Swift\_Current America/Tegucigalpa

America/Thule America/Thunder\_Bay America/Tijuana

America/Toronto America/Tortola America/Vancouver

America/Virgin America/Whitehorse America/Winnipeg

America/Yakutat America/Yellowknife

**Antarctica**

Antarctica/Casey Antarctica/Davis Antarctica/DumontDUrville Antarctica/Macquarie Antarctica/Mawson

Antarctica/McMurdo Antarctica/Palmer Antarctica/Rothera Antarctica/South\_Pole Antarctica/Syowa

Antarctica/Vostok

**Arctic**

Arctic/Longyearbyen

**Asia**

Asia/Aden Asia/Almaty Asia/Amman Asia/Anadyr Asia/Aqtau

Asia/Aqtobe Asia/Ashgabat Asia/Ashkhabad Asia/Baghdad Asia/Bahrain

Asia/Baku Asia/Bangkok Asia/Beirut Asia/Bishkek Asia/Brunei

Asia/Calcutta Asia/Choibalsan Asia/Chongqing Asia/Chungking Asia/Colombo

Asia/Dacca Asia/Damascus Asia/Dhaka Asia/Dili Asia/Dubai

Asia/Dushanbe Asia/Gaza Asia/Harbin Asia/Hebron Asia/Ho\_Chi\_Minh

Asia/Hong\_Kong Asia/Hovd Asia/Irkutsk Asia/Istanbul Asia/Jakarta

Asia/Jayapura Asia/Jerusalem Asia/Kabul Asia/Kamchatka Asia/Karachi

Asia/Kashgar Asia/Kathmandu Asia/Katmandu Asia/Khandyga Asia/Kolkata

Asia/Krasnoyarsk Asia/Kuala\_Lumpur Asia/Kuching Asia/Kuwait Asia/Macao

Asia/Macau Asia/Magadan Asia/Makassar Asia/Manila Asia/Muscat

Asia/Nicosia Asia/Novokuznetsk Asia/Novosibirsk Asia/Omsk Asia/Oral

Asia/Phnom\_Penh Asia/Pontianak Asia/Pyongyang Asia/Qatar Asia/Qyzylorda

Asia/Rangoon Asia/Riyadh Asia/Saigon Asia/Sakhalin Asia/Samarkand

Asia/Seoul Asia/Shanghai Asia/Singapore Asia/Taipei Asia/Tashkent

Asia/Tbilisi Asia/Tehran Asia/Tel\_Aviv Asia/Thimbu Asia/Thimphu

Asia/Tokyo Asia/Ujung\_Pandang Asia/Ulaanbaatar Asia/Ulan\_Bator Asia/Urumqi

Asia/Ust-Nera Asia/Vientiane Asia/Vladivostok Asia/Yakutsk Asia/Yekaterinburg

Asia/Yerevan

**Atlantic**

Atlantic/Azores Atlantic/Bermuda Atlantic/Canary Atlantic/Cape\_Verde Atlantic/Faeroe

Atlantic/Faroe Atlantic/Jan\_Mayen Atlantic/Madeira Atlantic/Reykjavik Atlantic/South\_Georgia

Atlantic/St\_Helena Atlantic/Stanley

**Australia**

Australia/ACT Australia/Adelaide Australia/Brisbane Australia/Broken\_Hill Australia/Canberra

Australia/Currie Australia/Darwin Australia/Eucla Australia/Hobart Australia/LHI

Australia/Lindeman Australia/Lord\_Howe Australia/Melbourne Australia/North Australia/NSW

Australia/Perth Australia/Queensland Australia/South Australia/Sydney Australia/Tasmania

Australia/Victoria Australia/West Australia/Yancowinna

**Europe**

Europe/Amsterdam Europe/Andorra Europe/Athens Europe/Belfast Europe/Belgrade

Europe/Berlin Europe/Bratislava Europe/Brussels Europe/Bucharest Europe/Budapest

Europe/Busingen Europe/Chisinau Europe/Copenhagen Europe/Dublin Europe/Gibraltar

Europe/Guernsey Europe/Helsinki Europe/Isle\_of\_Man Europe/Istanbul Europe/Jersey

Europe/Kaliningrad Europe/Kiev Europe/Lisbon Europe/Ljubljana Europe/London

Europe/Luxembourg Europe/Madrid Europe/Malta Europe/Mariehamn Europe/Minsk

Europe/Monaco Europe/Moscow Europe/Nicosia Europe/Oslo Europe/Paris

Europe/Podgorica Europe/Prague Europe/Riga Europe/Rome Europe/Samara

Europe/San\_Marino Europe/Sarajevo Europe/Simferopol Europe/Skopje Europe/Sofia

Europe/Stockholm Europe/Tallinn Europe/Tirane Europe/Tiraspol Europe/Uzhgorod

Europe/Vaduz Europe/Vatican Europe/Vienna Europe/Vilnius Europe/Volgograd

Europe/Warsaw Europe/Zagreb Europe/Zaporozhye Europe/Zurich

**Indian**

Indian/Antananarivo Indian/Chagos Indian/Christmas Indian/Cocos Indian/Comoro

Indian/Kerguelen Indian/Mahe Indian/Maldives Indian/Mauritius Indian/Mayotte

Indian/Reunion

**Pacific**

Pacific/Apia Pacific/Auckland Pacific/Chatham Pacific/Chuuk Pacific/Easter

Pacific/Efate Pacific/Enderbury Pacific/Fakaofo Pacific/Fiji Pacific/Funafuti

Pacific/Galapagos Pacific/Gambier Pacific/Guadalcanal Pacific/Guam Pacific/Honolulu

Pacific/Johnston Pacific/Kiritimati Pacific/Kosrae Pacific/Kwajalein Pacific/Majuro

Pacific/Marquesas Pacific/Midway Pacific/Nauru Pacific/Niue Pacific/Norfolk

Pacific/Noumea Pacific/Pago\_Pago Pacific/Palau Pacific/Pitcairn Pacific/Pohnpei

Pacific/Ponape Pacific/Port\_Moresby Pacific/Rarotonga Pacific/Saipan Pacific/Samoa

Pacific/Tahiti Pacific/Tarawa Pacific/Tongatapu Pacific/Truk Pacific/Wake

Pacific/Wallis Pacific/Yap