# ClickBlocks\Utils\DT

## General information

| | |
|---|---|
| **Inheritance** | DateTime |
| **Subclasses** | no |
| **Interfaces** | DateTimeInterface |
| **Source** | Framework/utils/dt.php |

**DT** class is designed for different manipulations with date and time. Using this class you can receive information considering various components of date and time, convert date into different formats, operate date validation, modify date, work with time zones. This whole new class doesn't require additional extensions, it contains wide range of static methods which make it so easy to use.

## Public static methods

### isDate()

```
public static boolean isDate(string $date, string $format = null)
```

| | | |
|---|---|---|
| **$data** | string | string representing the date. |
| **$format** | string | format of the given date. |

Validates the specified date format. Method returns TRUE if the given string is a correct date formatted according to the specified format and FALSE otherwise. If the date format is not specified the method will try to match the given date with one of the possible formats. Example:

```
echo (int)DT::isDate('2012-10-23 10:12:09', 'Y-m-d H:i:s'); // will output 1.
echo (int)DT::isDate('20121015091201', 'YmdHis');           // will output 1.
echo (int)DT::isDate('234524234');                          // will output 0.
echo (int)DT::isDate('2014, 12 Mon 12:34');                 // will output 1.
```

### isLeapYear()

```
public static boolean isLeapYear(integer $year = null)
```

| | | |
|---|---|---|
| **$year** | integer | the year, that should be checked for a leap. |

Checks whether the given year is a leap year. If no year is specified the current year will be checked. Example:

```
echo (int)DT::isLeapYear(2000); // will output 1.
echo (int)DT::isLeapYear(2014); // will output 0.
```

### getInfo()

```
public static array getInfo(string $date, string $format)
```

| | | |
|---|---|---|
| **$data** | string | string representing the date. |
| **$format** | string | format of the given date. |

Returns associative array with detailed info about given date. Example:

```
print_r(DT::getInfo('Thu, 13 Feb 2014 06:43:53 +0100', 'D, d M Y H:i:s O'));
```

The above example will output:

```
Array
(
    [year] => 2014
    [month] => 2
    [day] => 13
    [hour] => 6
    [minute] => 43
    [second] => 53
    [fraction] =>
    [warning_count] => 0
    [warnings] => Array
        (
        )

    [error_count] => 0
    [errors] => Array
        (
        )

    [is_localtime] => 1
    [zone_type] => 1
    [zone] => -60
    [is_dst] =>
    [relative] => Array
        (
            [year] => 0
            [month] => 0
            [day] => 0
            [hour] => 0
            [minute] => 0
            [second] => 0
            [weekday] => 4
        )

)
```

## getHour()

```
public static integer getHour(string $date, string $format)
```

| | | |
|---|---|---|
| **$data** | string | string representing the date. |
| **$format** | string | format of the given date. |

Returns value of the hour component of the given date.

## getMinute()

```
public static integer getMinute(string $date, string $format)
```

| | | |
|---|---|---|
| **$data** | string | string representing the date. |
| **$format** | string | format of the given date. |

Returns value of the minute component of the given date.

### getSecond()

```
public static integer getSecond(string $date, string $format)
```

| | | |
|---|---|---|
| **$data** | string | string representing the date. |
| **$format** | string | format of the given date. |

Returns value of the second component of the given date.

### getFraction()

```
public static integer getFraction(string $date, string $format)
```

| | | |
|---|---|---|
| **$data** | string | string representing the date. |
| **$format** | string | format of the given date. |

Returns value of the fraction component of the given date.

### getDay()

```
public static integer getDay(string $date, string $format)
```

| | | |
|---|---|---|
| **$data** | string | string representing the date. |
| **$format** | string | format of the given date. |

Returns value of the day component of the given date.

### getMonth()

```
public static integer getMonth(string $date, string $format)
```

| | | |
|---|---|---|
| **$data** | string | string representing the date. |
| **$format** | string | format of the given date. |

Returns value of the month component of the given date.

### getYear()

```
public static integer getYear(string $date, string $format)
```

| | | |
|---|---|---|
| **$data** | string | string representing the date. |
| **$format** | string | format of the given date. |

Returns value of the year component of the given date.

### getMonthDays()

```
public static integer|boolean getMonthDays(integer $year, integer $month)
```

| | | |
|---|---|---|
| **$year** | integer | some valid year. |

| | | |
|---|---|---|
| **month** | integer | some valid month of the given year. |

Returns number of days at the specified month of the given year. This method return FALSE if the given year and month isn't valid. Example:

```
echo DT::getMonthDays(2014, 2); // will output 28
echo DT::getMonthDays(2119, 6); // will output 30
echo (int)DT::getMonthDays(2000, 13); // will output 0
```

### timestamp()

```
public static integer|boolean timestamp(string $date, string $format = null)
```

| | | |
|---|---|---|
| **$data** | string | string representing the date. |
| **$format** | string | format of the given date. |

Returns unix timestamp for the given date formatted to the specified format. If the date format is not specified the method will try to parse the date from one of the suitable formats. The method returns FALSE if the given date is not valid. Example:

```
echo DT::timestamp('2013-02-01 02:13:14', 'Y-m-d H:i:s'); // will output 1359681194
echo DT::timestamp('2013-02-01 02:13:14');                // will output 1359681194
echo (int)DT::timestamp('24543223');                      // will output 0
```

### date2date()

```
public static string|boolean date2date(string $date, string $out, string $in = null)
```

| | | |
|---|---|---|
| **$date** | string | string representing the date. |
| **$out** | string | the output format of the date. |
| **$in** | string | the input format of the date. |

Converts the given date from one format to another. If the input date format is not specified the method will try to parse the date from one of the suitable formats. The method returns FALSE if the given date is not valid. Example:

```
echo DT::date2date('2013-02-01 02:13:14', 'd m y h:i sa', 'Y-m-d H:i:s'); // will output 01 02 13 02:13 14am
echo DT::date2date('2013-02-01 02:13:14', 'd m y h:i sa');               // will output 01 02 13 02:13 14am
echo (int)DT::date2date('23542434', 'd m y h:i sa');                     // will output 0
```

### date2sql()

```
public static string|boolean date2sql(string $date, string $format = null, boolean $shortFormat = false)
```

| | | |
|---|---|---|
| **$date** | string | string representing the date. |
| **$format** | string | the input date format. |
| **$shortFormat** | boolean | determines whether the short MySQL date format ("Y-m-d") will be used as the output date format. |

Converts the given date from the specified format to the MySQL date format ("Y-m-d" or "Y-m-d H:i:s"). If the input date format is not specified the method will try to parse the date from one of the suitable formats. The method returns FALSE if the given date is not valid. Example:

```
echo DT::date2sql('Thu, 13 Feb 2014 06:43:53 +0100', 'D, d M Y H:i:s O'); // will output 2014-02-13 06:43:53
echo DT::date2sql('Thu, 13 Feb 2014 06:43:53 +0100', null, true);         // will output 2014-02-13
echo (int)DT::date2sql('23542434');                                       // will output 0
```

### sql2date()

```
public static string|boolean sql2date(string $date, string $format, boolean $shortFormat = false)
```

| | | |
|---|---|---|
| **$date** | string | string representing the date. |
| **$format** | string | the output date format. |
| **$shortFormat** | string | determines whether the short MySQL date format ("Y-m-d") is used as the input date format. |

Converts the given date from the MySQL date format ("Y-m-d" or "Y-m-d H:i:s") to the specified format. The method returns FALSE if the given date is not valid.

```
echo DT::sql2date('2014-02-13 06:43:53', 'D, d M Y H:i:s O'); // will output Thu, 13 Feb 2014 06:43:53 +0100
echo DT::sql2date('2014-02-13', 'D, d M Y', true);            // will output Thu, 13 Feb 2014
```

### date2atom()

```
public static string|boolean date2atom(string $date, string $format = null)
```

| | | |
|---|---|---|
| **$date** | string | string representing the date. |
| **$format** | string | the input date format. |

Converts the given date from specified format to the atom date format. If the input date format is not specified the method will try to parse the date from one of the suitable formats. The method returns FALSE if the given date is not valid.

### atom2date()

```
public static string|boolean atom2date(string $date, string $format)
```

| | | |
|---|---|---|
| **$date** | string | string representing the date. |
| **$format** | string | the output date format. |

Converts the given date formatted from the atom date format to the given format. The method returns FALSE if the given date is not valid.

### date2rss()

```
public static string|boolean date2rss(string $date, string $format = null)
```

| | | |
|---|---|---|
| **$date** | string | string representing the date. |
| **$format** | string | the input date format. |

Converts the given date from specified format to the RSS date format. If the input date format is not specified the method will try to parse the date from one of the suitable formats. The method returns FALSE if the given date is not valid.

### rss2date()

```
public static string|boolean rss2date(string $date, string $format)
```

| | | |
|---|---|---|
| **$date** | string | string representing the date. |
| **$format** | string | the output date format. |

Converts the given date from RSS date format to the given format. The method returns FALSE if the given date is not valid.

## date2cookie()

```
public static string|boolean date2cookie(string $date, string $format = null)
```

| | | |
|---|---|---|
| **$date** | string | string representing the date. |
| **$format** | string | the input date format. |

Converts the given date from specified format to the cookie date format. If the input date format is not specified the method will try to parse the date from one of the suitable formats. The method returns FALSE if the given date is not valid.

## cookie2date()

```
public static string|boolean cookie2date(string $date, string $format)
```

| | | |
|---|---|---|
| **$date** | string | string representing the date. |
| **$format** | string | the output date format. |

Converts the given date from cookie date format to the given format. The method returns FALSE if the given date is not valid.

## compare()

```
public static integer|boolean compare(string $date1, string $date2, $format = null)
```

| | | |
|---|---|---|
| **$date1** | string | string representing the first date. |
| **$date2** | string | string representing the second date. |
| **$format** | string | format of the given dates. |

Compares two given dates.
The method returns 1 if the second date larger than the first date.
The method returns -1 if the second date smaller than the first date.
The method returns 0 if the both dates are equal.
The method returns FALSE if the given dates are not valid.
Example:

```
echo DT::compare('2001-01-02', '2001-05-10'); // will output 1
echo DT::compare('2001-05-10', '2001-01-02'); // will output -1
echo DT::compare('2001-01-02', '2001-01-02'); // will output 0
```

## difference()

```
public static string|boolean difference(string $date1, string $date2, string $format = null, string $component = '%r%a')
```

| | | |
|---|---|---|
| **$date1** | string | string representing the first date ("from" date). |
| **$date2** | string | string representing the second date ("to" date). |
| **$format** | string | format of the given dates. |
| **$component** | string | the date component. |

Computes the difference between two dates given in the same format within the date components. The method returns difference between dates in

days by default. The method returns FALSE if the given dates are not valid. Example:

```
echo DT::difference('2001-01-01', '2001-05-10');               // will output 129
echo DT::difference('2001-01-01', '2001-05-10', 'Y-d-m', '%d'); // will output 4
```

## countdown()

```
public static array countdown(string $date1, string $date2, string $format = null)
```

| | | |
|---|---|---|
| **$date1** | string | string representing the first date ("from" date). |
| **$date2** | string | string representing the second date ("to" date). |
| **$format** | string | format of the given dates. |

Calculates the difference between the two dates given in the same format. If the specific date format is not set the method will try to parse dates into one of the suitable formats. The method returns array of the following elements: second, minute, hour, day and returns FALSE if the given dates are not valid. Example:

```
print_r(DT::countdown('2001-01-01 10:15:31', '2001-05-10 04:12:05', 'Y-d-m H:i:s'));
```

The above example will output:

```
Array
(
    [day] => 276
    [hour] => 16
    [minute] => 56
    [second] => 34
)
```

## past()

```
public static string past(string $date, string $now, string $format = null)
```

| | | |
|---|---|---|
| **$date** | string | the start date. |
| **$now** | string | the current date. |

Returns text representation of the time elapsed since the specified date. Example:

```
echo DT::past('2001-01-01 10:15:31', '2002-05-10 04:12:05', 'Y-d-m H:i:s') . PHP_EOL;
echo DT::past('2001-01-01 10:15:31', '2001-05-10 04:12:05', 'Y-d-m H:i:s') . PHP_EOL;
echo DT::past('2001-01-01 10:15:31', '2001-05-01 04:12:05', 'Y-d-m H:i:s') . PHP_EOL;
echo DT::past('2001-01-01 10:15:31', '2001-01-01 10:17:05', 'Y-d-m H:i:s') . PHP_EOL;
echo DT::past('2001-01-01 10:15:31', '2001-01-01 10:15:58', 'Y-d-m H:i:s') . PHP_EOL;
echo DT::past('2001-01-01 10:15:31', '2001-01-01 10:15:31', 'Y-d-m H:i:s');
```

The above example will output:

```
a year ago
9 months ago
3 days ago
a minute ago
27 seconds ago
just now
```

## now()

```
public static string now(string $format, string|DateTimeZone $timezone = null)
```

| | | |
|---|---|---|
| **$format** | string | the output format of the current date. |
| **$timezone** | string, DateTimeZone | the timezone of the current date. |

Returns the current date of the specific format.

```
echo DT::now('Y-m-d H:i:s');                // will output 2014-02-13 13:02:49
echo DT::now('Y-m-d H:i:s', 'Pacific/Fiji'); // will output 2014-02-14 00:02:49
```

### tomorrow()

```
public static string tomorrow(string $format, string|DateTimeZone $timezone = null)
```

| | | |
|---|---|---|
| **$format** | string | the output format of the tomorrow's date. |
| **$timezone** | string, DateTimeZone | the timezone of the tomorrow's date. |

Returns the tomorrow's date of the specific format.

```
echo DT::tomorrow('Y-m-d H:i:s');                // will output 2014-02-14 13:38:35
echo DT::tomorrow('Y-m-d H:i:s', 'Pacific/Fiji'); // will output 2014-02-15 00:38:35
```

### yesterday()

```
public static string yesterday(string $format, string|DateTimeZone $timezone = null)
```

| | | |
|---|---|---|
| **$format** | string | the output format of the yesterday's date. |
| **$timezone** | string, DateTimeZone | the timezone of the yesterday's date. |

Returns the yesterday's date of the specific format.

```
echo DT::yesterday('Y-m-d H:i:s');                // will output 2014-02-12 13:43:18
echo DT::yesterday('Y-m-d H:i:s', 'Pacific/Fiji'); // will output 2014-02-13 00:43:18
```

### getDefaultTimeZone()

```
public static string getDefaultTimeZone()
```

Returns the default timezone used by all date/time functions.

### setDEfaultTimeZone()

```
public static boolean setDefaultTimeZone(string $timezone)
```

| | | |
|---|---|---|
| **$timezone** | string | the new default timezone. |

Sets the default timezone used by all date/time functions in a script. The method returns FALSE if the timezone isn't valid, or TRUE otherwise.

### getTimeZoneAbbreviation()

```
public static string getTimeZoneAbbreviation(string|DateTimeZone $timezone)
```

| | | |
|---|---|---|
| **$timezone** | string, DateTimeZone | the given timezone. |

Returns abbreviation of the given time zone. The time zone can be an instance of `DateTimeZone` or a string. Example:

```
echo DT::getTimeZoneAbbreviation('Europe/Berlin'); // will output CET
```

## getTimeZoneList()

```
public static array getTimeZoneList(boolean $combine = false, integer $what = DateTimeZone::ALL, string $country = null)
```

| | | |
|---|---|---|
| **$combine** | boolean | determines whether the method returns associative array or not. |
| **$what** | integer | one of `DateTimeZone` class constants. |
| **$country** | string | a two-letter ISO 3166-1 compatible country code. This argument is only used when **$what** is set to `DateTimeZone::PER_COUNTRY.` |

Returns numerical index array with all timezone identifiers. If **$combine** is TRUE the method returns associated array with keys which are time zones. Example:

```
print_r(Utils\DT::getTimeZoneList(false, \DateTimeZone::ANTARCTICA));
print_r(Utils\DT::getTimeZoneList(true, \DateTimeZone::PER_COUNTRY, 'AU'));
```

The above script will output:

```
Array
(
    [0] => Antarctica/Casey
    [1] => Antarctica/Davis
    [2] => Antarctica/DumontDUrville
    [3] => Antarctica/Macquarie
    [4] => Antarctica/Mawson
    [5] => Antarctica/McMurdo
    [6] => Antarctica/Palmer
    [7] => Antarctica/Rothera
    [8] => Antarctica/South_Pole
    [9] => Antarctica/Syowa
    [10] => Antarctica/Vostok
)
Array
(
    [Antarctica/Macquarie] => Antarctica/Macquarie
    [Australia/Adelaide] => Australia/Adelaide
    [Australia/Brisbane] => Australia/Brisbane
    [Australia/Broken_Hill] => Australia/Broken_Hill
    [Australia/Currie] => Australia/Currie
    [Australia/Darwin] => Australia/Darwin
    [Australia/Eucla] => Australia/Eucla
    [Australia/Hobart] => Australia/Hobart
    [Australia/Lindeman] => Australia/Lindeman
    [Australia/Lord_Howe] => Australia/Lord_Howe
    [Australia/Melbourne] => Australia/Melbourne
    [Australia/Perth] => Australia/Perth
    [Australia/Sydney] => Australia/Sydney
)
```

## getGMTTimeZoneList()

```
public static array getGMTTimeZoneList(array $replacement = null, integer $what = DateTimeZone::ALL, string $country = null)
```

| | | |
|---|---|---|
| **$replacement** | array | can be used to replace some timezone names by others. |

| | | |
|---|---|---|
| **$what** | integer | one of \DateTimeZone class constants. |
| **$country** | string | a two-letter ISO 3166-1 compatible country code. This argument is only used when **$what** is set to `DateTimeZone::PER_COUNTRY`. |

Returns list (associative array) of timezones in GMT format. Example:

```
print_r(Utils\DT::getGMTTimeZoneList(null, \DateTimeZone::ANTARCTICA));
print_r(Utils\DT::getGMTTimeZoneList(['Australia/Darwin' => 'AU/Drw'], \DateTimeZone::PER_COUNTRY, 'AU'));
```

The above script will output:

```
Array
(
    [Antarctica/Palmer] => (GMT-03:00) Palmer
    [Antarctica/Rothera] => (GMT-03:00) Rothera
    [Antarctica/Syowa] => (GMT+03:00) Syowa
    [Antarctica/Mawson] => (GMT+05:00) Mawson
    [Antarctica/Davis] => (GMT+07:00) Davis
    [Antarctica/Casey] => (GMT+08:00) Casey
    [Antarctica/DumontDUrville] => (GMT+10:00) DumontDUrville
    [Antarctica/Macquarie] => (GMT+11:00) Macquarie
    [Antarctica/McMurdo] => (GMT+13:00) McMurdo
    [Antarctica/South_Pole] => (GMT+13:00) South Pole
)
Array
(
    [Australia/Eucla] => (GMT+08:45) Eucla
    [Australia/Perth] => (GMT+08:00) Perth
    [Australia/Darwin] => (GMT+09:30) AU/Drw
    [Australia/Adelaide] => (GMT+10:30) Adelaide
    [Australia/Brisbane] => (GMT+10:00) Brisbane
    [Australia/Broken_Hill] => (GMT+10:30) Broken Hill
    [Australia/Lindeman] => (GMT+10:00) Lindeman
    [Australia/Currie] => (GMT+11:00) Currie
    [Australia/Hobart] => (GMT+11:00) Hobart
    [Australia/Lord_Howe] => (GMT+11:00) Lord Howe
    [Antarctica/Macquarie] => (GMT+11:00) Macquarie
    [Australia/Melbourne] => (GMT+11:00) Melbourne
)
```

## getGMTTimeZone()

```
public static string getGMTTimeZone(string|DateTimeZone $timezone)
```

| | | |
|---|---|---|
| **$timezone** | string, DateTimeZone | name of the given timezone or timezone object. |

Returns timezone string in GMT format. Example:

```
echo DT::getGMTTimeZone('Australia/Lord_Howe'); // will output (GMT+11:00) Lord Howe
```

## zone2zone()

```
public static string zone2zone(string $date, string|DateTimeZone $zoneOut, string|DateTimeZone $zoneIn, string $out, string $in =
```

| | | |
|---|---|---|
| **$date** | string | the given date. |
| **$zoneOut** | string, DateTimeZone | the output timezone of the date. |
| **$zoneIn** | string, DateTimeZone | the input timezone of the date. |

| | | |
|---|---|---|
| **$out** | string | the output date format. |
| **$in** | string | the input date format. |

Converts the given date from one timezone and date format to another timezone and date format. Example:

```
echo Utils\DT::zone2zone('2014-03-10 12:35:17', 'Antarctica/Vostok', 'America/Dominica',  \DateTime::RSS, 'Y-d-m H:i:s');
// will output Fri, 03 Oct 2014 22:35:17 +0600
```

# Public non-static methods

## __construct()

```
public void __construct(string|DateTimeInterface $date = 'now', string $format = null, string|DateTimeZone $timezone = null)
```

| | | |
|---|---|---|
| **$date** | string, DateTimeInterface | the given date. |
| **$format** | string | format of the given date. |
| **$timezone** | string, DateTimeZone | timezone of the given date. |

Class constructor. If **$date** is not specified, the current date will be taken. If **$format** is not set any suitable format will be used. If **$timezone** is not defined the default timezone will be taken as timezone of the given date.

## __toString()

```
public string __toString()
```

Returns string representation of the datetime object in RFC 2822 format. This method is automatically invoked when an datetime object converts to string.

## setTimezone()

```
public self|boolean setTimezone(string|DateTimeZone $timezone)
```

| | | |
|---|---|---|
| **$timezone** | string, DateTimeZone | timezone of a datetime object that associated with the class instance. |

Sets the timezone for the internal datatime object. The method returns the object itself for method chaining or FALSE on failure.

## getTimestamp()

```
public integer getTimestamp(string|DateTimeZone $timezone = null)
```

| | | |
|---|---|---|
| **$timezone** | string, DateTimeZone | timezone of a datetime object that associated with the class instance. |

Returns the Unix timestamp representing the date. Example:

```
$dt = new DT();
echo $dt->getTimestamp() . PHP_EOL;
echo $dt->getTimestamp('Pacific/Nauru');
```

The above script will output:

```
1392643927
1392683527
```

## format()

```
public string format(string $format, string|DateTimeZone $timezone = null)
```

| | | |
|---|---|---|
| **$format** | string | the output date format. |
| **$timezone** | string, DateTimeZone | the timezone of the output date. |

Returns the date formatted to the given format for the specified timezone. Example:

```
$dt = new DT('now');
echo $dt->format('Y-m-d H:i:s') . PHP_EOL;
echo $dt->format('Y-m-d H:i:s', 'Pacific/Nauru');
```

The above example will output:

```
2014-02-17 14:42:55
2014-02-18 01:42:55
```

## add()

```
public self|boolean add(string|DateInterval $interval)
```

| | | |
|---|---|---|
| **$interval** | string, DateInterval | a date interval. See more details here. |

Adds an amount of days, months, years, hours, minutes and seconds to a datetime object. The method returns the object itself for method chaining or FALSE on failure. Example:

```
$dt = new DT('2000-01-15 12:01:15');
$dt->add('10 days 1 hour 177 seconds');
echo $dt->format('Y-m-d H:i:s');
```

The above example will output:

```
2000-01-25 13:04:12
```

## sub()

```
public self|boolean sub(string|DateInterval $interval)
```

| | | |
|---|---|---|
| **$interval** | string, DateInterval | a date interval. See more details here. |

Subtracts an amount of days, months, years, hours, minutes and seconds from a datetime object. The method returns the object itself for method chaining or FALSE on failure. Example:

```
$dt = new DT('2000-01-15 12:01:15');
$dt->sub('10 days 1 hour 177 seconds');
echo $dt->format('Y-m-d H:i:s');
```

The above example will output:

```
2000-01-05 10:58:18
```

## addDay()

```
public self addDay(integer $day = 1)
```

| | | |
|---|---|---|
| **$day** | integer | the date increment in days. |

Adds an amount of days to the datetime object. The amount of days can be negative. The method returns the object itself for method chaining or FALSE on failure.

### addMonth()

```
public self addMonth(integer $month = 1)
```

| | | |
|---|---|---|
| **$month** | integer | the date increment in months. |

Adds an amount of months to the datetime object. The amount of months can be negative. The method returns the object itself for method chaining or FALSE on failure.

### addYear()

```
public self addYear(integer $year = 1)
```

| | | |
|---|---|---|
| **$years** | integer | the date increment in years. |

Adds an amount of years to the datetime object. The amount of years can be negative. The method returns the object itself for method chaining or FALSE on failure.

### addHour()

```
public self addHour(integer $hour = 1)
```

| | | |
|---|---|---|
| **$hour** | integer | the date increment in hours. |

Adds an amount of hours to the datetime object. The amount of hours can be negative. The method returns the object itself for method chaining or FALSE on failure.

### addMinute()

```
public self addMinute(integer $minute = 1)
```

| | | |
|---|---|---|
| **$minute** | integer | the date increment in minutes. |

Adds an amount of minutes to the datetime object. The amount of minutes can be negative. The method returns the object itself for method chaining or FALSE on failure.

### addSeconds()

```
public self addSecond(integer $second = 1)
```

| | | |
|---|---|---|
| **$second** | integer | the date increment in seconds. |

Adds an amount of seconds to the datetime object. The amount of seconds can be negative. The method returns the object itself for method chaining or FALSE on failure.

### diff()

```
public DateInterval|boolean diff(mixed $date, boolean $absolute = false)
```

| | | |
|---|---|---|
| **$date** | mixed | the date to compare to. |
| **$absolute** | boolean | determines whether to return absolute difference. |

Returns `DateInterval` object representing the difference between the two dates or FALSE on failure. Example:

```
$dt1 = new DT('2009-10-11');
$dt2 = new DT('2009-10-13');
echo $dt1->diff($dt2)->format('%R%a days');
```

The above script will output:

```
+2 days
```

## cmp()

```
public integer cmp(mixed $date, string|DateTimeZone $timezone = null)
```

| | | |
|---|---|---|
| **$date** | mixed | the date to compare to. |
| **$timezone** | string, DateTimeZone | timezone of the given dates. |

Compares two given dates.
The method returns 1 if the second date larger than the first date.
The method returns -1 if the second date smaller than the first date.
The method returns 0 if the both dates are equal.
Example:

```
$dt1 = new DT('2009-10-11 10:15:37', 'Y-m-d H:i:s', 'America/Belize');
$dt2 = new DT('2009-10-12 01:12:13', 'Y-m-d H:i:s', 'Indian/Mahe');
echo $dt1->cmp($dt2, 'Europe/Moscow'); // will output -1 ($dt1 > $dt2 for Moscow timezone)
echo $dt2->cmp($dt1, 'Europe/Moscow'); // will output 1 ($dt2 < $dt1 for Moscow timezone)
echo $dt1->cmp($dt1, 'Europe/Moscow'); // will output 0 ($dt1 == $dt2)
```

## getPeriod()

```
public Generator getPeriod(string|DateInterval $interval, integer|DateTimeInterface $end, boolean $excludeStartDate = false)
```

| | | |
|---|---|---|
| **$interval** | string, DateInterval | the interval between recurrences within the period. |
| **$end** | integer, DateTimeInterface | the number of recurrences or the end date of the period. |
| **$excludeStartDate** | boolean | determines whether the start date is excluded from the set of recurring dates within the period. |

The method allows iteration over a set of dates and times, recurring at regular intervals, over a given period. It returns a Generator object representing of the given date period. Example:

```
$dt = new Utils\DT('now', null, 'America/Belize');
$period = $dt->getPeriod(new \DateInterval('P7D'), 5, true);
foreach ($period as $dt)
{
  echo $dt->format('Y-m-d H:i:s', 'Europe/Moscow') . PHP_EOL;
}
```

The above example will output:

```
2014-02-24 20:36:35
2014-03-03 20:36:35
2014-03-10 20:36:35
2014-03-17 20:36:35
2014-03-24 20:36:35
```