How do I compare two DateTime objects in PHP 5.2.8?

Asked 15 years, 6 months ago Modified 3 years, 5 months ago Viewed 442k times

Part of PHP Collective

359

Having a look on the PHP documentation, the following two methods of the `DateTime` object would both seem to solve my problem:

- DateTime::diff : Get the difference and use that to determine which is more ancient.
- DateTime::getTimestamp : Get the UNIX timestampt and compare those.

Both these methods are marked in the doco as being available in version >= 5.3 (and, not surprisingly, if I try to call them I find they don't exist). I can't find any specific documentation for 5.2.8 so I am not sure if there are equivalent methods in my version. I have Googled the problem and found an eclectic range of solutions, none of which answer my very simple requirements:

- How do I compare two DateTime objects?
- Where can I find the doco for previous PHP versions? Specifically version 5.2.8?

For some context, I have the following code:

```
$st_dt = new DateTime(verifyParam ('start_date'));
$end_dt = new DateTime(verifyParam ('end_date'));

// is the end date more ancient than the start date?
if ($end_dt < $start_dt)
```

Apparently there is no comparison operator on this guy.

Edit

Apparently my assumptions were completely false (thanks Milen for illustrating this so effectively). There is a comparison operator and it works just fine thanks. Sometimes I really miss a compiler. The bug is in the code above, I am sure you will find it much faster than I did :).

PHP php datetime

Share

Improve this question

Follow

edited May 27, 2019 at 14:18

halfer

20.4k ● 19 ● 108 ● 200

asked Jun 7, 2009 at 2:33

RedBlueThing

42.5k ● 17 ● 97 ● 122

2 Regarding the lack of compiler - set "error_reporting" to "E_ALL" and you'll get notices like "Notice: Undefined variable: start_dt in ...". – Milen A. Radev Jun 8, 2009 at 9:05

5 Also, please, use htmlentities on your `$_POST` vars, or kitten shall be killed. – Clement Herreman Aug 30, 2010 at 8:56

2 And where is the error? :p , I'm doing it too U_U . Thanks in advance! – castarco Nov 8, 2011 at 12:34

2 @castarco I initialise $st_dt , but I compare against an uninitialised $start_dt. Check your variable names and perhaps follow Milen's suggestion and set error_reporting to E_ALL to get undefined variable warnings. :) – RedBlueThing Nov 8, 2011 at 22:20

8 Answers

Sorted by: Highest score (default)

512

The following seems to confirm that there are comparison operators for the DateTime class:

```
dev:~# php
<?php
date_default_timezone_set('Europe/London');

$d1 = new DateTime('2008-08-03 14:52:10');
$d2 = new DateTime('2008-01-03 11:11:10');
var_dump($d1 == $d2);
var_dump($d1 > $d2);
var_dump($d1 < $d2);
?>
bool(false)
bool(true)
bool(false)
dev:~# php -v
PHP 5.2.6-1+lenny3 with Suhosin-Patch 0.9.6.2 (cli) (built: Apr 26 2009
20:09:03)
Copyright (c) 1997-2008 The PHP Group
Zend Engine v2.2.0, Copyright (c) 1998-2008 Zend Technologies
dev:~#
```

Share Improve this answer Follow

answered Jun 7, 2009 at 3:04

Milen A. Radev

62.4k ● 22 ● 107 ● 110

10 Thanks Milen, looks like I just needed my false assumptions removed and suddenly the glaring bug in my code became obvious to me. – RedBlueThing Jun 7, 2009 at 5:26

3 Hmm, this is interesting. Maybe at some point we'll be able to overload operators in user-defined classes. – Ionuț G. Stan Jun 7, 2009 at 11:23

1 From php.net/manual/en/language.operators.comparison.php Built-in classes can define its own comparison, different classes are uncomparable, same class - compare properties the same way as arrays (PHP 4), PHP 5 has its own explanation – Saul Oct 27, 2010 at 11:17

11 watch out when comparing a datetime with no hour set and one with it set(default constructor) – max4ever Mar 19, 2012 at 16:45

1 TiMESPLiNTER, I believe the warning is that if you're only interested in comparing dates you might overlook the fact that the hours are different, and thus two `DateTime`s with the same date will not be compared as equal when you believe they should. You can remedy this by explicitly setting the time components of the object to zero before comparing them. – Jason Apr 21, 2015 at 15:01

102

From the official documentation:

> As of PHP 5.2.2, DateTime objects can be compared using comparison operators.

```php
$date1 = new DateTime("now");
$date2 = new DateTime("tomorrow");

var_dump($date1 == $date2); // false
var_dump($date1 < $date2); // true
var_dump($date1 > $date2); // false
```

For PHP versions before 5.2.2 (actually for any version), you can use diff.

```php
$datetime1 = new DateTime('2009-10-11'); // 11 October 2013
$datetime2 = new DateTime('2009-10-13'); // 13 October 2013

$interval = $datetime1->diff($datetime2);
echo $interval->format('%R%a days'); // +2 days
```

Share

Improve this answer

Follow

edited Jan 4, 2018 at 2:40

Pang

10.1k ● 146 ● 85 ● 124

answered Feb 22, 2016 at 11:22

Roberto Alonso

1,201 ● 1 ● 8 ● 9

9 I find this answer best since it quotes the manual instead of just checking the behavior and assuming the outcome is as expected. SO is not a place for guesswork. Bravo Roberto. – cprn Sep 19, 2016 at 14:18

4 @roberto DateTime::diff has only been added in PHP 5.3 – NeXuS May 16, 2017 at 4:19

35

You can also compare epoch seconds :

```
$d1->format('U') < $d2->format('U')
```

Source : http://laughingmeme.org/2007/02/27/looking-at-php5s-datetime-and-datetimezone/ (quite interesting article about DateTime)

Share Improve this answer Follow

answered Nov 4, 2012 at 20:33

Julien
9,432 ● 10 ● 67 ● 87

19 Beware that `format` produces a *string*, so that's a string comparison. It's barely a problem after 1000000000 epoch time (roughly Sep 9th, 2001), but if you have to deal with dates *before* that, you may incur into problems due to different number lengths. Either convert the results to numbers (subtracting them works too), or use a truly sortable format like `c`. – MaxArt Jun 18, 2015 at 8:25

1 @MaxArt could you please elaborate on the problems due to different string lengths? The manual, regarding strings in numeric contexts: "If the string does not contain any of the characters '.', 'e', or 'E' and the numeric value fits into integer type limits (as defined by PHP_INT_MAX), the string will be evaluated as an integer. In all other cases it will be evaluated as a float." Future dates (around 2038) can overflow signed 32-bit integers, but what's the issue with older dates? – Adrian Günter Sep 15, 2015 at 18:45

1 @AdrianGünter The problem is that we wouldn't be in a numeric context, but we'd deal with *strings*. Numeric, but still strings. So a *string* comparison would be made. – MaxArt Sep 16, 2015 at 9:28

2 @MaxArt: as per the documentation: If you compare a number with a string or the comparison involves numerical strings, then each string is converted to a number and the comparison performed numerically. These rules also apply to the switch statement. The type conversion does not take place when the comparison is === or !== as this involves comparing the type as well as the value.. – Frédéric Marchal Apr 15, 2016 at 9:09

2 @FrédéricMarchal Yes, I indeed learnt later that my previous comment is in fact false. PHP has a really wicked way to treat these cases, no other language (as I know of) casts to numbers when you have two strings to compare. It really screws your code if you actually want to make a lexicographical comparison. – MaxArt Apr 15, 2016 at 14:53

25

If you want to compare dates and not time, you could use this:

```
$d1->format("Y-m-d") == $d2->format("Y-m-d")
```

Share

Improve this answer

Follow

edited Nov 26, 2013 at 14:15

krsteeve
1,804 ● 4 ● 20 ● 29

answered Nov 26, 2013 at 13:49

blablabla
1,478 ● 16 ● 17

6 You can also set reset time. `$d1->setTime(0, 0, 0);` – Athlan Aug 20, 2014 at 15:58

2

```php
$elapsed = '2592000';
// Time in the past
$time_past = '2014-07-16 11:35:33';
$time_past = strtotime($time_past);

// Add a month to that time
$time_past = $time_past + $elapsed;

// Time NOW
$time_now = time();

// Check if its been a month since time past
if($time_past > $time_now){
    echo 'Hasnt been a month';
}else{
    echo 'Been longer than a month';
}
```

Share Improve this answer Follow

answered Jul 17, 2014 at 13:57

Kyle Coots
2,121 ● 1 ● 18 ● 24

0

You mentioned `DateTime::diff` to compare two dates, you can use it to get a more precise value than just a boolean.

For example the number of days:

```php
$date1 = new DateTime('2021-07-09');
$date2 = new DateTime('2020-10-21');

echo $date1->diff($date2)->format('%R%a days');
```

But you could also use an alias of `DateTime::diff` which is `date_diff` *(PHP 5 >= 5.3.0, PHP 7, PHP 8)*

```php
$date1 = new DateTime('2021-07-09');
$date2 = new DateTime('2020-10-21');

echo date_diff($date1, $date2)->format('%R%a days');
```

Share Improve this answer Follow

answered Jul 9, 2021 at 8:59

Dylan KAS
5,640 ● 2 ● 21 ● 38

This may help you.

```
$today = date("m-d-Y H:i:s");
$thisMonth =date("m");
$thisYear = date("y");
$expectedDate = ($thisMonth+1)."-08-$thisYear 23:58:00";


if (strtotime($expectedDate) > strtotime($today)) {
    echo "Expected date is greater then current date";
    return ;
} else
{
 echo "Expected date is lesser then current date";
}
```

Share

Improve this answer

Follow

edited Jan 4, 2018 at 2:41

Pang

10.1k ● 146 ● 85 ● 124

answered Feb 8, 2014 at 8:29

Tarun Gupta

6,393 ● 2 ● 44 ● 39

@SteelBrain do you think that timestamp limitation bother above code which is having all current date time, Please read the code again, It basically check for the guy $expectedDate which will always be in the current month. I dont think so we should think about the timestamp limitation here in this case. – Tarun Gupta Aug 4, 2014 at 9:58

I know, but it's not a recommended way, (I didn't downvote :-)) – Steel Brain Aug 4, 2014 at 13:19

Can you suggest a recommended way. – Tarun Gupta Aug 5, 2014 at 6:39

sure, `$today = new DateTime("now");` `$time = DateTime::createFromFormat('d-m-Y',"26-October-1998");` `if ($today > $time){echo "today is greater";}else{echo "other time is greater";}` cheers. – Steel Brain Aug 5, 2014 at 7:14

As of PHP 7.x, you can use the following:

```
$aDate = new \DateTime('@'.(time()));
$bDate = new \DateTime('@'.(time() - 3600));

$aDate <=> $bDate; // => 1, `$aDate` is newer than `$bDate`
```

Share Improve this answer Follow

answered Mar 3, 2018 at 13:43

jens1o

755 ● 7 ● 14

1 This answer needs explanation of its unusual notation [backslash and '@'] and how it works [concat '@' with result of time(),] – Bilbo Nov 12, 2020 at 19:18

4 Answers need to include explanation of any unusual notation, like backslash, At-symbol, and spaceship operator. Code should include comments about how works, like: - concat '@' with result of time(), optionally subtracting one hour's worth of seconds - instantiate DateTime objects - compare DateTime objects, returning 0 if equal, or 1 if left value is greater, or -1 if right value is greater [Comment editing timeout window expired; cannot delete previous comment] – Bilbo Nov 12, 2020 at 19:27