How to compare DateTime in C#?

Ask Question

112

19

I don't want user to give the back date or time.

How can I compare if the entered date and time is LESS then the current time?

If the current date and Time is 17-Jun-2010 , 12:25 PM , I want user cannot give date before 17 Jun -2010 and time before 12:25 PM.

Like my function return false if the time entered by user is 16-Jun-2010 and time 12:24 PM

c# .net datetime

edited Sep 9 '14 at 7:01
Kiquenet
7,162 27 110 199

asked Jun 17 '10 at 6:57
Ashish Ashu
5,363 34 73 107

6 Answers

Home

PUBLIC

Stack Overflow

Tags

Users

Jobs

151 ✓

MSDN: DateTime.Compare

```
DateTime date1 = new DateTime(2009, 8, 1, 0, 0, 0);
DateTime date2 = new DateTime(2009, 8, 1, 12, 0, 0);
int result = DateTime.Compare(date1, date2);
string relationship;

if (result < 0)
   relationship = "is earlier than";
else if (result == 0)
   relationship = "is the same time as";
else
   relationship = "is later than";

Console.WriteLine("{0} {1} {2}", date1, relationship, date2);
// The example displays the following output:
//    8/1/2009 12:00:00 AM is earlier than 8/1/2009 12:00:00 PM
```

answered Jun 17 '10 at 6:59
Ahmet Kakıcı
5,384 4 31 46

2 Your answer provides a way to see what the difference is instead of just knowing that the date is before or after. Of course his answer is better for the OP, but yours is better for some people who got here from google (self included). – levininja Apr 14 '17 at 18:46

1 I found your answer valuable because I'm looking at some legacy code where this is used – Erik Bergstedt Jul 3 '17 at 7:16

Here is the MSDN link – Harsh Raj Jun 29 '18 at 9:40

But from debug point of view, MuSTaNG's answer is more readable. – Himalaya Garg Sep 14 '18 at 8:04

From MSDN about `DateTime.Compare` :

0

> To determine the relationship of t1 to t2, the Compare method compares the Ticks property of t1 and t2 but **ignores their Kind property**. Before comparing DateTime objects, ensure that the objects represent times in the same time zone.

This is relevant for operators like `>`, `<` and `==`. Thus, a simple comparison may give an unexpected result when dealing with `DateTimes` that are represented in different timezones.

In general case you need to compare `DateTimes` with the same `Kind`:

```
if (date1.ToUniversalTime() < date2.ToUniversalTime())
    Console.WriteLine("date1 is earlier than date2");
```

answered Mar 15 at 15:03

AlbertK

3,569 1 11 19

-3

```
public static bool CompareDateTimes(this DateTime firstDate, DateTim
{
   return firstDate.Day == secondDate.Day && firstDate.Month == seco
firstDate.Year == secondDate.Year;
}
```

edited Jul 31 '18 at 9:07

Jesse de Bruijne

2,551 6 14 28

answered Jul 31 '18 at 8:27

Majid Bajwa

1 1

3 In .NET, "Compare" usually means "relative comparison" and not "equality check". The question here was about relative comparison. Additionally you made no effort to format the code properly. – Lasse Vågsæther Karlsen Jul 31 '18 at 8:30

17

MuSTaNG's answer says it all, but I am still adding it to make it a little more elaborate, with links and all.

The conventional operators

- *greater than* (`>`),
- *less than* (`<`),
- *equality* (`==`),
- and more

are available for `DateTime` since .NET Framework 1.1. Also, addition and subtraction of `DateTime` objects are also possible using conventional operators `+` and `-` .

One example from MSDN:

Equality:

```
System.DateTime april19 = new DateTime(2001, 4, 19);
System.DateTime otherDate = new DateTime(1991, 6, 5);

// areEqual gets false.
bool areEqual = april19 == otherDate;

otherDate = new DateTime(2001, 4, 19);
// areEqual gets true.
areEqual = april19 == otherDate;
```

Other operators can be used likewise.

Here is the list all operators available for `DateTime` .

edited May 23 '17 at 12:03

Community ♦
1 1

answered Oct 25 '16 at 6:55

Sнаđошƒаӽ
7,445 10 45 67

1 You could just edit their answer to elaborate. In any case, you cannot use `+` for two `DateTime` operands, You can do `DateTime - DateTime` , or `DateTime + TimeSpan` , or `DateTime - TimeSpan` . – Joey Oct 25 '16 at 7:05

1

```
//Datetime compare.
private int CompareTime(string t1, string t2)
{
    TimeSpan s1 = TimeSpan.Parse(t1);
    TimeSpan s2 = TimeSpan.Parse(t2);
    return s2.CompareTo(s1);
}
```

edited Apr 21 '17 at 13:55

Sнаđошƒаӽ
7,445 10 45 67

answered Nov 4 '16 at 6:07

Phan Đình Diệu
11 3

He want to compare `DateTime` not `string` – Danh Nov 4 '16 at 6:26

you can convert datetime to string – Phan Đình Diệu Nov 21 '16 at 8:24

or: you can use: DateTime.Compare(startDate, endDate) – Phan Đình Diệu Nov 21 '16 at 9:52

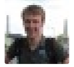
Not my problem and is it difference from Ahmet's? – Danh Nov 21 '16 at 9:58

276

Microsoft has also implemented the operators '<' and '>'. So you use these to compare two dates.

```
if (date1 < DateTime.Now)
   Console.WriteLine("Less than the current time!");
```

answered Jun 17 '10 at 7:19



MuSTaNG

3,219 2 10 6

6 Source MSDN; these are documeted as DateTime Operators awkwardly labelled as "DateTime.Greater than", "DateTime.LessThanOrEqualTo"..... msdn.microsoft.com/en-us/library/ff986512%28v=vs.90%29.aspx – Salman Siddiqui Jan 23 '15 at 18:09

current .NET version: msdn.microsoft.com/en-us/library/ff986512(v=vs.110).aspx – juFo Sep 21 '16 at 14:54

I'm using Unity 2017 and using this operator to sort lists gives me wrong results. I even tried to compare DateTime.ticks directly and that failed, too. I had to use DateTime.CompareTo to arrive at right results and I don't know why. – lin Jan 4 '18 at 9:08

1 This is wrong. This can't compare UTC and Local Time in the right way. – Altiano Gerung May 24 '18 at 13:12

1 This is not complete answer. When taking DateTime Kind to ocnsideration, it should be either `date1.ToLocalTime() < DateTime.Now` or `date1.ToUniversalTime() < DateTime.UtcNow` . – yurislav Sep 27 '18 at 11:46