How to set time to midnight for current day?

55

2

Every time that I create a non-nullable datetime in my mvc3 application it defaults to now(), where now is current date with current time. I would like to default it to today's date with 12am as the time.

I'm trying to default the time in my mvc...but...the following isn't setting to todays date @12am. Instead it defaults to now with current date and time.

```
private DateTime _Begin = new DateTime(DateTime.Now.Year, DateTime.Now.Month,
DateTime.Now.Day, 12, 0, 0);
public DateTime Begin { get { return _Begin; } set { _Begin = value; } }
```

How can I set to 12am for the current date for non-nullable datetime?

c# .net

edited Nov 20 '12 at 5:55
Erik Philips
42.4k 6 97 131

asked Nov 20 '12 at 5:42
genxgeek
5,557 28 107 189

6 Answers

90

You can use the `Date` property of the DateTime object - eg

```
DateTime midnight = DateTime.Now.Date;
```

So your code example becomes

PS. going back to your original code setting the hours to **12** will give you time of *noon* for the current day, so instead you could have used **0**...

```
var now = DateTime.Now;
new DateTime(now.Year, now.Month, now.Day, 0, 0, 0);
```

edited Mar 23 '15 at 16:36
CCowan
54 7

answered Nov 20 '12 at 5:44
Chris Moutray
15k 6 35 62

13 There is also `DateTime.Today` – Erv Walter Nov 20 '12 at 5:46

2 @Erv Indeed if you reflect (disassemble the M$ dll) you'll find `Today` is simply a wrapper for `DateTime.Now.Date` - so personally I find this just shows a little more intent in my code... – Chris Moutray Nov 20 '12 at 5:47 ✎

2 It made me chuckle to think `Now.Date` is *overly* verbose for an extra 3 characters... – Chris Moutray Nov 20 '12 at 6:06 ✎

10 About the last line: **Never** use many calls to the `Now` property getter in the same expression. `Now` changes! If the code runs near a time where `Day` (and possibly `Month` and `Year`) changes, it will give a wrong output. Like the first of the `Now` -s could be "Nov. 30th, 23:59:59.9999...", and the last one could be "Dec. 1st, 00:00:00.0000...". Your result could then be **November 1st**, way off. – Jeppe Stig Nielsen Nov 20 '12 at 8:56 ✎

1 @JeppeStigNielsen very good point! – Chris Moutray Nov 20 '12 at 9:01

25

I believe you are looking for `DateTime.Today` . The documentation states:

> An object that is set to today's date, with the time component set to 00:00:00.

http://msdn.microsoft.com/en-us/library/system.datetime.today.aspx

Your code would be

```
DateTime _Begin = DateTime.Today;
```

it's not the answer but it helps me :-) – vNext Apr 4 '16 at 4:55

1 @vNext Yes it is, it answers the question exactly and is the most concise answer. – Jamie Kitson Mar 8 at 9:45

1

Using some of the above recommendations, the following function and code is working for search a date range:

Set date with the time component set to 00:00:00

```
public static DateTime GetDateZeroTime(DateTime date)
{
    return new DateTime(date.Year, date.Month, date.Day, 0, 0, 0);
}
```

Usage

```
var modifieddatebegin = Tools.Utilities.GetDateZeroTime(form.modifieddatebegin);

var modifieddateend = Tools.Utilities.GetDateZeroTime(form.modifieddateend.AddDays(1));
```

answered Oct 10 '16 at 16:12

Ravi Ram
12.7k 18 63 90

Your GetDateZeroTime implementation is equivalent to accessing the Date property of the DateTime object, i.e. date.Date – djoyce Nov 2 '17 at 10:20

1

Only need to set it to

```
DateTime.Now.Date
```

```
Console.WriteLine(DateTime.Now.Date.ToString("yyyy-MM-dd HH:mm:ss"));
```

> "2017-04-08 00:00:00"

on my machine.

edited Dec 6 '18 at 10:21
croxy
3,064 7 21 40

answered Apr 9 '17 at 1:06
Reno
19 2

0

Most of the suggested solutions can cause a 1 day error depending on the time associated with each date. If you are looking for an integer number of calendar days between to dates, regardless of the time associated with each date, I have found that this works well:

```
return (dateOne.Value.Date - dateTwo.Value.Date).Days;
```

edited Jan 5 '18 at 3:21
jwpfox
4,145 10 37 41

answered Jan 5 '18 at 3:02
Rusty West
1

-1

Try this:

```
DateTime Date = DateTime.Now.AddHours(-DateTime.Now.Hour).AddMinutes(-
DateTime.Now.Minute)
   .AddSeconds(-DateTime.Now.Second);
```

Output will be like:

> 07/29/2015 00:00:00

edited Dec 6 '18 at 10:23

answered Jul 29 '15 at 6:38