Visual Studio 2008 Visual Studio 2005 Visual Studio 2010 Intermediate Dev Visual Studio .NET C#

# C# Date Time Parser

**Sergiy Stoyan**

4.72/5 (53 votes)

Feb 11, 2009 Public Domain 4 min read 259167 2519

Parsing date and (or) time from a string

**Download source**
**Download latest version from SourceForge**

## Introduction

Once I needed to detect and recognize date and/or time contained in internet messages. Those messages were sent by different users, and so could not comply with certain format. They consisted usually of 1 or 2 sentences and it was impossible to foresee where a date and/or time is within them. Thus, built-in .NET method `DateTime.Parse()` could not help because this method can parse only `string`s consisting exactly of a date/time presentation, it cannot look for date/time among text. Moreover, some yet prevalent date presentations appeared to not be recognized by `DateTime.Parse()`.

What I needed was a C# function like the universal Perl's `str2time()` or PHP's `strtotime()`. After some Googling, I was surprised not to find such a code in C#. That's why I wrote a date/time parsing class presented here.

## Description

The class `DateTimeRoutines` exposes several date/time parsing methods. The general idea is finding first instance of the date and/or time within a `string` and converting it into `DateTime`.

| Method | Description |
|---|---|
| `TryParseDateTime()` | Tries to find both a date and a time within the passed `string`. If a date or time was not found, it returns `false`. |
| `TryParseDate()` | Tries to find a date within the passed `string`. It always returns time 0:0:0. If year of the date was not found then it accepts, by default, the current year; this rule can be changed by specifying `DefaultDate`. If a date was not found, it returns `false`. |
| `TryParseTime()` | Tries to find a time within the passed `string`. If a date previously found in the `string` is specified as a parameter, then it looks for a time around this date. It always returns date 1/1/1. If a time was not found, it returns `false`. |
| `TryParseDateOrTime()` | Tries to find a date and/or time within the passed `string`. If only a time was found then, by default, date is set by today's date; this rule can be changed by specifying `DefaultDate`. If neither date nor time was found, it returns `false`. |

These methods accept `DateTimeFormat` parameter that specifies a recognition format used as preferred in ambivalent instances.

These methods return a `ParsedDateTime` object. This object describes whether a date (or time) was found within the `string` and where it was found (if it was), and also hosts a `DateTime` structure as a result of parsing.

Also, these methods have derivations that return a `DateTime` directly instead of `ParsedDateTime`. The use of the origin methods is preferable though because their output allows knowing whether a date (or time) was really found or if it was set by default value.

Notice that `TryParseDateTime()` and `TryParseTime()` may return different times in the case where a `string` contains more than one time. `TryParseDateTime()` looks for a time around a date, while `TryParseTime()` returns a time that was found first.

## Local and Absolute Time

`ParsedDateTime.DateTime` is always considered local meaning that it reflects the parsed `string` literally. If UTC offset or time zone abbreviation indicating that the time is absolute was found in the time string then `ParsedDateTime.IsUtcOffsetFound` is `true` and `ParsedDateTime.UtcDateTime` is UTC date&time. If `ParsedDateTime.IsUtcOffsetFound` is `false`, then `ParsedDateTime.UtcDateTime` should be disregarded as undefined.

Notice that `TryParseDate()` does not detect time zone

## Usage

The date formats that can be recognized by `DateTimeRoutines` can be seen in the test strings listed below (the complete list of parsed formats can be found in `Test` project supplied with the code):

Plain Text

```
@"Member since:      10-Feb-2008"
@"Last Update: 18:16 11 Feb '08 "
@"date    Tue, Feb 10, 2008 at 11:06 AM"
@"see at 12/31/2007 14:16:32"
@"sack finish 14:16:32 November 15 2008, 1-144 app"
@"Genesis Message - Wed 04 Feb 08 - 19:40"
@"The day 07/31/07 14:16:32 is "
@"Shipping is on us until December 24, 2008 within the U.S."
@" 2008 within the U.S. at 14:16:32"
@"5th November, 1994, 8:15:30 pm"
@"7 boxes January 31 , 14:16:32."
@"the blue sky of Sept  30th  2008 14:16:32"
@" e.g. 1997-07-16T19:20:30+01:00"
@"Apr 1st, 2008 14:16:32 tufa 6767"
@"wait for 07/31/07 14:16:32"
@"later 12.31.08 and before 1.01.09"
@"Expires: Sept  30th  2008 14:16:32"
@"Offer expires Apr 1st, 2007, 14:16:32"
@"Expires  14:16:32 January 31,"
```

```
@"Expires  14:16:32 January 31-st."
@"Expires 23rd January 2010."
@"Expires January 22nd, 2010."
@"Expires DEC 22, 2010."
```

A code sample if you need to get only date:

```C#
string str = @"The last round was June 10, 2005; this time the unbroken record was held.";
DateTimeRoutines.ParsedDateTime pdt;
if (DateTimeRoutines.TryParseDate(str, DateTimeRoutines.DateTimeFormat.USA_DATE, out pdt))
    Console.WriteLine("Date was found: " + pdt.DateTime.ToString());
```

A code sample if you want to get date and, if possible, time:

```C#
string str = @"The last round was June 10, 2005; this time the unbroken record was held.";
DateTimeRoutines.ParsedDateTime pdt;
if (DateTimeRoutines.TryParse(str, DateTimeRoutines.DateTimeFormat.USA_DATE, out pdt)
    && pdt.IsDateFound
    )
    Console.WriteLine("Date was found: " + pdt.DateTime.ToString());
```

A code sample if you want to get only completely specified date and time:

```C#
string str = @"The last round was June 10, 2005 10:30AM; this time the unbroken record was held.";
DateTimeRoutines.ParsedDateTime pdt;
if(str.TryParseDateTime(DateTimeRoutines.DateTimeFormat.USA_DATE, out pdt))
    Console.WriteLine("Date&time was found: " + pdt.DateTime.ToString());
```

A code sample if you want to get UTC date and time:

```C#
string str = @"Your program recognizes string : 21 Jun 2010 04:20:19 -0430 blah blah.";
DateTimeRoutines.ParsedDateTime pdt;
if(str.TryParseDateTime(DateTimeRoutines.DateTimeFormat.USA_DATE, out pdt) && pdt.IsUtcOffsetFound)
Console.WriteLine("UTC date&time was found: " + pdt.UtcDateTime.ToString());
```

## .NET Version Consistency

`DateTimeRoutines` is formed as a .NET 4 DLL that exposes the parsing methods as extensions for `string` class. The DLL can be called by .NET 2 code just as well. However, if you want to embed `DateTimeRoutines` source code into your .NET 2 project, you'll have to remove keyword `this` from the method parameters.

## Conclusion

This code satisfied my needs. I did not want to implement too wide a recognition capability like say, the one provided by Perl's `str2time()`, because more wide recognition capability results in a higher error rate when the parser tries to detect a date/time within any part of a `string`.

Nevertheless, `DateTimeRoutines` is capable of recognizing the formats that usually are used in a correspondence. If you find some prevalent date/time format which is not recognized, please let me know and I'll update the code.

## The Code

In the attached code, you can find the `DateTimeRoutines` project containing:

- A class `DateTimeRoutines` that is compiled as a DLL
- Project `Test`

The code is licensed as public domain code.

The latest version can be found on SourceForge.

Be happy!

## History

- 11th February, 2009
  - Initial post
- 14th February, 2009
  - `TryParseDateTime()` added
- 18th December, 2009
  - `TryParseDate()` updated
- 3rd March, 2010
  - `TryParseDate()` updated
- 12th March, 2010
  - `locks` removed
- 13th March, 2010
  - Formed as a DLL
  - Methods formed as extensions for `string` class
  - `TryParse()` renamed to `TryParseDateOrTime()`
- 13th July, 2010
  - Updated source code
- 15th May, 2011
  - Fixed 12pm and 12am
  - Upgraded to C# 4.0
- 18th April, 2012
  - Added one more date format

- 28th June, 2012
  - Added UTC recognition;