FoxDates

Date and Time Functions for Visual FoxPro

Project Manager: Rick Borup

To use, instantiate *clsFoxDates* from *foxdates.prg* and call the desired method.

Examples:

```
oFoxDates = NEWOBJECT( "clsFoxDates", "foxdates.prg")
oFoxDates.GetLastOfMonth( {^2019-11-12})  && returns 11/30/2019
oFoxDates.GetLastOfMonth( {^2019-02-01})  && returns 02/28/2019 (not a leap year)
oFoxDates.GetLastOfMonth( {^2020-02-01})  && returns 02/29/2020 (is a leap year)
```

If you make changes, run the FoxUnit test suite in clsFoxDatesTest.prg to ensure all tests still pass. Note: some functions in this class depend on other functions in this class, so changes to one function can affect others.

GetFirstOfMonth()

Pass a date, get back the first day of that month.

```
oFoxDates.GetFirstOfMonth( {^2019-11-19})    && returns 11/01/2019
```

GetLastOfMonth()

Pass a date, get back the last day of that month.

```
oFoxDates.GetLastOfMonth( {^2019-11-19})    && returns 11/30/2019
```

GetDaysInMonth()

Pass a date, get back the number of days in that month.

```
oFoxDates.GetDaysInMonth( {^2019-11-19})    && returns 30
```

GetLastEOM()

Pass a date, get back the last day of the previous month.

```
oFoxDates.GetLastEOM( {^2019-11-19})    && returns 10/31/2019
```

GetBOQ()

Pass a date, get back the first day of the calendar quarter.

```
oFoxDates.GetBOQ( {^2019-11-19})    && returns 10/01/2019
```

GetEOQ()

Pass a date, get back the last date of the calendar quarter.

```
oFoxDates.GetEOQ( {^2019-11-19})    && returns 12/31/2019
```

GetLastEOQ()

Pass a date, get back the date of the preceding end of quarter.

```
oFoxDates.GetLastEOQ( {^2019-11-19})    && returns 09/30/2019
```

GetLastEOY()

Pass a date, get back the date of the preceding end of year.

```
oFoxDates.GetLastEOY( {^2019-11-19})    && returns 12/31/2018
```

GetLastMonday()

Pass a date, get back the date of the preceding Monday.

```
oFoxDates.GetLastMonday( {^2019-11-19})    && returns 11/18/2019
```

GetNextMonday()

Pass a date, get back the date of the next Monday.

```
oFoxDates.GetNextMonday( {^2019-11-19})    && returns 11/25/2019
```

GetDateFromString()

Pass a date as a string in mm/dd/yyyy or similar format, get it back as a VFP date.

```
oFoxDates.GetDateFromString( "11/19/2019")    && returns 11/19/2019 as a VFP date
oFoxDates.GetDateFromString( "11-19-2019")    && returns 11/19/2019 as a VFP date
oFoxDates.GetDateFromString( "11.19.2019")    && returns 11/19/2019 as a VFP date
```

IsLeapYear()

Pass a date, find out whether it's a leap year.

```
oFoxDates.IsLeapYear( {^2019-11-19})    && returns .F.
oFoxDates.IsLeapYear( {^2020-11-19})    && returns .T.
```

GetDateDayOrdinal()

Pass a date, get back the day of the month as an ordinal value like "first", "tenth", "nineteenth", or "thirty-first".

```
oFoxDates.GetDateDayOrdinal( {^2019-11-19})    && returns "nineteenth"
```

GetFormattedDateString()

Pass a date, get back a string formatted for display.

```
oFoxDates.GetFormattedDateString( {^2019-11-19}, 1)    && returns "November 19, 2019"
oFoxDates.GetFormattedDateString( {^2019-11-19}, 2)    && returns "Tuesday, November 19, 2019"
```

GetNthBusinessDay()

Pass the month, the year, and the desired business day, get back the date.

```
oFoxDates.GetNthBusinessDay( 11, 2019, 10)    && returns 11/14/2019 (the 10th business day)
```

IsHoliday()

Pass a date and an optional country code, find out if it's a holiday. Country code defaults to USA. The only other option at this time is Canada.

```
oFoxDates.IsHoliday( {^2019-11-19})    && returns .F.
oFoxDates.IsHoliday( {^2019-11-28})    && returns .T. (Thanksgiving Day in the USA)
oFoxDates.IsHoliday( {^2019-11-28}, "Canada")    && returns .F. (not Thanksgiving Day in Canada)
oFoxDates.IsHoliday( {^2019-10-14}, "Canada")    && returns .T. (is Thanksgiving Day in Canada)
```

GetTimeString()

Pass numeric values for hours and minutes, get back a string formatted as a time.

```
oFoxDates.GetTimeString( 5, 11)    && returns "05:11"
```

GetDisplayTime()

Pass a time as a string like hh:mm, get back a string that includes AM or PM.

```
oFoxDates.GetDisplayTime( "05:11")    && returns "05:11 AM"
oFoxDates.GetDisplayTime( "17:11")    && returns "05:11 PM"
```

GetSecondsFromTimeString()

Pass a time as a string like hh:mm, get back the number of seconds since midnight.

```
oFoxDates.GetSecondsFromTimeString( "05:11")    && returns 18660.00
```

GetTimeStringFromSeconds()

Pass the number of seconds since midnight, get back a time string like hh:mm.

```
oFoxDates.GetTimeStringFromSeconds( 18660.00)    && returns "05:11"
```

GetEndTime()

Pass a starting time and a duration, get back the ending time.

```
oFoxDates.GetEndTime( "05:11", 30)    && returns "05:41"
```

GetDuration()

Pass a start time and an end time, get back the duration in minutes.

```
oFoxDates.GetDuration( "05:11", "05:41")    && returns 30.0000
```

GetRFC2822()

Pass a date or a datetime, get back a string in RFC 2822 format.

Note - does NOT adjust for time zones. All times are assumed to be UTC (+0000) unless an offset string is passed as the second parameter.

```
oFoxDates.GetRFC2822( {^2019-11-19})    && returns "Tue, 19 Nov 2019 12:00:00 +0000"
oFoxDates.GetRFC2822( {^2019-11-19 17:11:00})    && returns "Tue, 19 Nov 2019 17:11:00 +0000"
oFoxDates.GetRFC2822( DATETIME(), "-0600")    && returned "Tue, 19 Nov 2019 13:12:02 -0600"
```

IsValidTimeString()

Pass a time string, find out if it conforms to a valid time in hh:mm format.

```
oFoxDates.ValidateTimeString( "5pm")      && returns .F.
oFoxDates.ValidateTimeString( "05:11")    && returns .T.
```

Get24HourTimeString()

Pass a time in a common format like 10am, 1p, or 3:30pm and get back a string in 24-hour clock format as hhmm (no colon). Useful for storing time strings that can later be compared using VAL().

```
oFoxDates.Get24HourTimeString( "10am")    && returns "1000"
oFoxDates.Get24HourTimeString( "1p")      && returns "1300"
oFoxDates.Get24HourTimeString( "3:30pm")  && returns "1530"
```

GetIntervalDays()

Pass two dates, get back the number of days in the interval between them. Optional third parameter determines if the result is a semi-closed interval (default - includes the start date but not the end date), a closed interval (includes both dates), or an open interval (does not include either date).

```
oFoxDates.GetIntervalDays( DATE(), DATE() + 1)       && returns 1
oFoxDates.GetIntervalDays( DATE(), DATE() + 1, 0)    && returns 1
oFoxDates.GetIntervalDays( DATE(), DATE() + 1, 1)    && returns 2
oFoxDates.GetIntervalDays( DATE(), DATE() + 1, 2)    && returns 0
```