 Jewish Calendar Library

*JewishCalendar* is an open source .NET library for Jewish calendar functions and calculations.

In the JewishCalendar library, a single day of the Jewish calendar is represented by an instance of the ***JewishDate*** class.

There are 2 versions of the *JewishDate* class, JewishDate and JewishDateMicro:

* JewishCalendar.JewishDate *-* Based on the standard .NET class System.Globalization.HebrewCalendar, which was found to be very efficient.

1. This class can only represent dates for the years 5343 through 5999. This is because System.Globalization.HebrewCalendar is limited to those dates.
2. This class cannot be used for projects that do not have access to System.Globalization.HebrewCalendar. So for projects such as a .NET Micro Framework projects etc., you will need to use the alternative class, JewishDateMicro. You will also need to remove (or comment out) the file *JewishDate.cs* before compiling.

Even though the JewishDate class’s underlying calculations are directly based on System.Globalization.HebrewCalendar, there is a major difference between the System.Globalization.HebrewCalendar class and the JewishDate class.

The System.Globalization.HebrewCalendar class has Tishrei as month number one, while JewishDate has Nissan as the first month.

We have found that having Tishrei as the first month becomes very confusing, as all the months after Adar do not have a set number. They will all get a different number, depending on whether or not the year is a leap year.

Additionally, calling Tishrei as month number one is inherently incorrect. The Torah specifically instructs us to call *Nissan* the first month. (See Ramban in his Drasha for Rosh Hashana).

So the JewishDate class has Nissan as month number one. All the months are always numbered correctly and if the year is a leap year, Adar Sheini will be month number 13.

* JewishCalendar.JewishDateMicro - This class is somewhat less efficient than the *JewishDate* class but can represent any Jewish date from the date of creation of the world (either JD 1-1-0001 or JD 7-1-0001) onwards. It also has no dependence on the System.Globalization.HebrewCalendar so it can be used for .NET Micro Framework projects etc.

Otherwise, the functionality of both classes are identical.

The following are the different constructors you can use to create an instance of the JewishDate or JewishDateMicro classes:

* JewishDate() – Empty constructor. Creates a new object based on the current system date.
* JewishDate(int year, int month, int day)– creates a new Jewish date for the given Jewish year, month and day.

For example:

iJewishDate jd = new JewishDate(5767, 7, 29);

This will give you an instance representing the 29th day of Tishrei in the year תשס"ז (5767).

* JewishDate(System.DateTime date)– Creates a Jewish date corresponding to the given secular date.

For example:

iJewishDate jd = new JewishDate(

new DateTime(2015, 10, 16));

This will get you a Jewish date object for the 3rd of Cheshvan 5776.

Note: as the location is not specified here, we cannot determine what time sunset is. So if the given time is after sunset, the Jewish date will be a Jewish Day too early.

* JewishDate(Location location) - Gives you a JewishDate for the current system date and time at the given *Location*. This constructor checks the time of sunset at the given location. If the current system time is after sunset, it adds a day to the JewishDate object it creates. This, of course, is because the Jewish Date begins at sunset, not at midnight.
* JewishDate(System.DateTime date, Location location) *­*– Creates a JewishDate for the given secular date and time at the given location. This constructor checks the time of sunset at the given location. If the given time is after sunset, it adds a day to the JewishDate object it creates. The Jewish Date begins at sunset, not at midnight.

Both JewishDate classes implement the *JewishCalendar.IJewishDate* interface which gives them the following properties:

* Day - The Day in the month for this Jewish Date.
* Month - The number of the Jewish Month for this Jewish Date. As in the Torah, Nissan is month number one.
* MonthName - The name of the current Jewish Month (in English).
* Year – The number of years since creation for this Jewish Date.
* DayOfWeek – The day of the week for this Jewish Date. Type: System.DayOfWeek.
* GregorianDate – A regular System.DateTime containing the secular date on this Jewish Date.

Important note: In the JewishDate class, if the location was supplied in the constructor and the time at object creation was between sunset and midnight for that location, the GregorianDate will be incorrect, as it will be the next day – which is only correct for the Jewish Date. To get the proper Secular Date, use the GetSecularDate function.

* AbsoluteDate - The number of days that elapsed since the theoretical Gregorian date Sunday, December 31, 1 BCE. Since there is no year 0 in the calendar, the year following 1 BCE is 1 CE. So, the Gregorian date January 1, 1 CE is absolute date number 1.

Note: This property is mostly used for internal calculations but can be used for very quick date calculations.

Additionally, both JewishDate classes also have the following functions:

* ToString() - Returns the Jewish date in the format: Adar 14, 5775.
* ToLongDateString() - Returns the Jewish date in the format: The 14th day of Adar, 5775.
* ToShortDateString() – Identical to ToString(), returns the Jewish date in the format: Adar 14, 5775.
* ToLongDateStringHeb() - Returns the Jewish date in the format:

יום חמישי כ"ט תשרי תשע"ה.

* ToShortDateStringHeb() - Returns the Jewish date in the format: כ"ו אלול תשע"ה.

Additionally, both JewishDate classes also define the following operator functions:

* iJewishDate jd1 == iJewishDate jd2 - Returns true if both iJewishDate objects have the same day, month and year. You can also use the Equals(obj jd2) method or the extension method IsSameDate(iJewishDate js) for the same purpose.
* iJewishDate jd1 != iJewishDate jd2 - Returns true if both iJewishDate objects do not have the same day, month and year.
* iJewishDate jd1 > iJewishDate jd2- Returns true if the current iJewishDate object (left side) is chronologically after the second iJewishDate object (the one on the right).
* iJewishDate jd1 < iJewishDate jd2- Returns true if the current iJewishDate object (left side) is chronologically before the second iJewishDate object (the one on the right).
* iJewishDate jd1 >= iJewishDate jd2- Returns true if the current iJewishDate object (left side) is not chronologically earlier than the second iJewishDate object (the one on the right).
* iJewishDate jd1 <= iJewishDate jd2- Returns true if the current iJewishDate object (left side) is not chronologically later than the second iJewishDate object (the one on the right).
* iJewishDate jd1 + int days– Adds the specified number of days to the current iJewishDate object (the one on the left) and returns it as new JewishDate object.
* iJewishDate jd1 - int days– Subtracts the specified number of days from the current iJewishDate object (the one on the left) and returns it as new JewishDate object.
* iJewishDate jd1 - iJewishDate j2– Gets the difference in days between two Jewish dates.