Trabajando con Fechas - DateTime

En este tutorial veremos cómo trabajar con el tipo `Datetime` , el cual nos permite trabajar con fechas. En el mundo laboral, que es básicamente lo importante, utilizamos fechas continuamente, por lo que es importante tener los siguientes conceptos claros.

Índice

1 - Creación del objeto Datetime

Para crear el objeto Datetime tenemos multitud de opciones como podemos observar:

```
public DateTime(long ticks);
public DateTime(long ticks, DateTimeKind kind);
public DateTime(int year, int month, int day);
public DateTime(int year, int month, int day, Calendar calendar);
public DateTime(int year, int month, int day, int hour, int minute, int second);
public DateTime(int year, int month, int day, int hour, int minute, int second, Da
public DateTime(int year, int month, int day, int hour, int minute, int second, Ca
public DateTime(int year, int month, int day, int hour, int minute, int second, in
public DateTime(int year, int month, int day, int hour, int minute, int second, in
public DateTime(int year, int month, int day, int hour, int minute, int second, in
public DateTime(int year, int month, int day, int hour, int minute, int second, in
```

Sin embargo dos de ellas son las mas comunes, ya que son las que utilizamos cuando queremos compara fechas, y cuando queremos comparar tiempo además de fechas.

```
public DateTime(int year, int month, int day);
public DateTime(int year, int month, int day, int hour, int minute, int second);
```

Por lo que para crea un objeto de fecha utilizaremos la siguiente sentencia:

```
DateTime testFecha = new DateTime(1989, 11, 2, 11, 15, 16);
```

Donde:

- 1989 es el año.
- 11 es el mes.
- 2 es el dia.
- 11 es la hora, en formato 24h, por lo tanto, las 11 de la mañana.

- 15 son los minutos.
- 16 son los segundos.

Otra opción para crear las fechas es utilizar los "ticks" o golpes de reloj, estos cuentan desde el 1 de enero de 1970.

```
public DateTime(long ticks);
```

Porque eligieron el 1 de enero de 1970. Bueno el motivo es que cuando estaban desarrollando unix neceistaban una fecha para empezar a contar y seleccionaron esa. No tiene ningún misterio ni ningún otro motivo oculto.

Finalmente, podemos convertir fehas desde un string, para el cual tenemos dos opciones.

```
DateTime ejemploFecha = Convert.ToDateTime("10/22/2015 12:10:15 PM");
DateTime ejemploFecha = DateTime.Parse("10/22/2015 12:10:15 PM");
```

Ambas opciones son completamente válidas aunque tienen pequeñas diferencias internamente, las dos convertirán el texto en una fecha.

Desafortunadamente el ejemplo que tenemos encima no funciona correctamente, tan solo si estamos en una máquina de estados unidos, donde el formato para las fechas es mes/dia; Si por el contrario estamos en una máquina configurada para europa, el formato correcto para una fecha es dia/mes. *DesconozcoSudamérica.

1.1 - Locaización con cultureinfo

Para evitar este problema tenemos el tipo `CultureInfo` que se encuentra dentro de `System.Globalization` el cual nos permite especificar el formato de una feha, basandonos en el país junto al idioma. El siguiente código nos generaria el tipo `CultureInfo` con español de españa.

```
CultureInfo cultureInfoES = new CultureInfo("es-SP");
```

Mientras que si lo que queremos es un país de Sudámerica como por ejemplo Argentina, seria el siguiente código

```
CultureInfo cultureInfoAR = new CultureInfo("es-AR");
```

Por lo que para el ejemplo visto anteriormente (mes/dia) tenemos que utliizar el `CultureInfo` de Estados unidos:

```
CultureInfo cultureInfoUS = new CultureInfo("en-us");
DateTime ejemploFecha = Convert.ToDateTime("10/22/2015 12:10:15 PM", cultureInfoUS
```

Nota muy importante: Si indicamos mal el `CultureInfo` , el programa crasheara y detendrá su ejecución así que hay que estar seguros con lo que hacemos.

1.2 - Propiedades del tipo Datetime

A raíz de la fecha anterior podemos obtener la mayoría de la información que comúnmente utilizamos, como pueden ser dia, mes, dia de la semana, et. Todo ello nos viene dentro del Tipo Datetime, por lo que no tenemos que crear ningún método para obtenerlo.

Aquí muestro una pequeña lista con algunos ejemplos:

```
int dia = fecha.Day; //Nos devuelve el dia del mes en número (1, 2...30, 31)
int mes = fecha.month; //Nos devuelve el número de mes
int year = fecha.Year; //Nos devuelve el año
int hora = fecha.Hour; //Devuelve la hora
int minuto = fecha.Minute; //devuelve el minuto
int segundo = fecha.Second; //devuelve los segundos.
string diaDeLaSemana = fecha.DayOfWeek; //Devuelve el dia de la semana con letra (
int diaDelyear = fecha.DayOfYear; //Devuelve en que dia del año estamos, con núme
DateTime tiempo= fecha.Date; //devuelve horas:minutos:segundos
```

1.3 - Métodos del tipo Datetime

como en el caso anterior, el propio tipo nos trae infinidad de métodos para ser ejecutados. entre los que podemos añadir dias, meses o incluso tiempo.

```
fecha.AddDays(1); //añadira un día a la fecha actual
fecha.AddMonths(1);// Añadira un mes a la fecha actual.
fecha.AddYears(1); //Añadira un año a la fecha actual.
```

Como vemos todos los ejemplos son de añadir, pero qué pasa, si lo que queremos es restar dias, para ello tambien tenemos que utilizar el método de añadir, solo que el valor que le pasamos será negativo.

```
fecha.AddDays(-1); //restará un día a la fecha actual
fecha.AddMonths(-1);// restará un mes a la fecha actual.
fecha.AddYears(-1); //restará un año a la fecha actual.
```

Por último si lo que queremos es añadir tiempo, tenemos que utilizar un tipo concreto.

1.4 - El tipo TimeSpan

Similar al caso anterior, solo que ahora en el constructor podemos tenemos las opciones del tiempo

```
public TimeSpan(long ticks);
public TimeSpan(int hours, int minutes, int seconds);
public TimeSpan(int days, int hours, int minutes, int seconds);
public TimeSpan(int days, int hours, int minutes, int seconds, int milliseconds);
```

Como observamos, para crear el tiempo `TimeSpan` podemos hacerlo desde días, hasta los milisegundos. y lo realizaremos de la siguiente forma:

```
DateTime fecha = new Datetime(2019, 01, 01);
//fecha tendrá el valor de 1 de enero de 2019 a las 00h 00m

TimeSpan tiempo = new TimeSpan (1, 5, 30, 5);
//creamos un objeto timespan con valor de 1 dia, 5 horas, 30 minutos, 5 segundos

DateTime fechaActualizada = fecha.Add(tiempo);
//Sumamos el tiempo a la fecha anterior
//Resultado: 2 de enero de 2019 a las 5:30 de la mañana.
```

2 - Comparación de fechas

Comparar fechas es muy importante en el mundo real y es algo que hay que tener muy claro. Por ejemplo comparamos fechas dentro de filtros o en consultas a la base de datos, o consultas **LINQ** como la que vimos en el post anterior.

Cuando comparamos lo podemos hacer de dos formas:

- Utilizar el método `DateTime.Compare(fecha1, fecha2)` dentro del tipo estático `DateTime` .
- Utilizar la propia feha para compararala con la segunda, fecha1.CompareTo(fecha2);

En ambos casos el resultado que nos devuelve es el mismo, nos devolverá un entero (int) que corresponde con lo siguiente:

- `menor que 0` si la primera fecha es menor que la segunda.
- `0` si ambas fechas son iguales
- `mayor que 0` si la primera fecha es mayor que la segunda.

```
DateTime fecha1 = new DateTime(1989, 11, 2);
DateTime fecha2 = new DateTime(1978, 4, 15);
int fechaResultado = DateTime.Compare(fecha1, fecha2);
//O indistintamente
int fechaResultado = fecha1.CompareTo(fecha2);

if (fechaResultado < 0)
{
    Console.WriteLine("La primera fecha es menor");
}
else if (fechaResultado == 0)
{
    Console.WriteLine("Las fechas son iguales");
}
else
{
```

```
    Console.WriteLine("La segunda fecha es menor");
}
```

3 - I

Fin én
que ue
fijar

Por or
def

```
DateTime fecha= new DateTime(1989, 11, 2, 11, 15, 16);
fecha.ToString(); // resultado: 02/11/1989 11:15:16
fecha.ToShortDateString(); //resultado: 02/11/1989
fecha.ToLongDateSTring(); //Resultado: Jueves 2 octubre 1989
fecha.ToShortTimeString(); //resultado: 11:15
```

Además de las opciones que nos trae por defecto, podemos crear nuestra propia versión utilizando el método `.ToString()` ya que si le pasamos un valor por parámetro el compilador lo traduce a lo que necesitamos.

Los ejemplos anteriores se pueden representar tansolo con `.ToString()` de la siguiente manera:

Pero de qué nos sirve poder imprimir algo que ya podemos imprimir anteriormente. El método .ToString() es mucho más potente, ya que podemos pasarle una combinación de carácteres para que muestre el formato tal y como lo necesitamos. por ejemplo `yyyy` se traduce al año, `MM` se traduce al mes.

Algo muy común en todas las aplicaciones es tener un log, los logs guardan fecha y hora con precisión de microsegundo. Podemos crear un mensaje partiendo de una fecha que el formato sea similar al de un log.

```
DateTime fecha= new DateTime(1989, 11, 2, 11, 15, 16, 123);
fecha.ToString(yyy-MM-ddThh:mm:ss.ms);
// resultado: 1989-01-11T11:15:16.123
```

Como podemos observar los caracteres como `-` o `.` tambíen salen representados en el resultado.

Además de estos, tenemos muchos más códigos para pasar en el método `.ToString()` posteriormente podemos utilizar cualquiera de los métodos disponibles en la clase **string**.

Finalmente una tabla con todo lo que el método `.ToString()` permite añadir, notese la diferencia entre mayúsculas y minúsculas.

Epecificación	Descripción	Salida
d	Fecha corta	02/11/1989
D	Fecha larga	jueves 2 Noviembre 1989
t	Tiempo corto	11:15
T	Tiempo largo	11:16:16
f	Fecha y hora completa corto	Jueves 2 Noviembre 1989 11:15
F	Fecha y hora completa largo	Jueves 2 Noviembre 1989 11:15:16
g/G	fecha y hora por defecto	02/11/1989 11:15
M	Día y mes	02-Nov
r	RFC 1123 fecha	Jue 02 Nov 1989 11:15:16 GTM
s	fecha y hora para ordenar	1989-11-02T11:15:16
u	tiempo universal, con timezone	1989-11-02T11:15:16z
Y	mes año	Noviembre 1989
dd	Día	2
ddd	día corto	Jue
dddd	día completo	Jueves
hh	hora con 2 digitos	11

HH	hora con 2 digitos formato 24h	23
mm	mintuo con 2 digitos	15
MM	mes	11
MMM	nombre mes corto	Nov
MMMM	nombre mes largo	Noviembre
ss	segundos	16
fff	milisegundos	123
tt	AM/PM	PM
yy	año con 2 digitos	89
yyyy	año con 4 digitos	1989