03

Formatos Simplificados de Datas e Horas

Transcrição

Começaremos a trabalhar com alguns formatos simplificados de **data** e **hora** utilizando as funções do `DateTime()` do C#.

Criaremos uma nova linha para imprimir uma data, entretanto queremos exibir essa data por **extenso**.

Usaremos a mesma variável `data`, que utilizamos anteriormente, e a exibiremos no console.

```
static void Main(string[] args)
{
    Debug.WriteLine(data.ToString("D"));
}
```

A letra **D** em maiúscula, nos dá o formato da data por extenso. Vamos rodar a aplicação para ver o resultado.

```
sábado, 23 de setembro de 2017
```

Podemos utilizar esse formato para imprimir documentos legais, jornais, etc. Se você quiser usar um formato diferente, por exemplo, a data sem o ano, usaremos a letra `m`.

```
static void Main(string[] args)
{
    Debug.WriteLine(data.ToString("D"));
    Debug.WriteLine(data.ToString("m"));
}
```

E como resultado, temos somente `23 de setembro` que é uma data por extenso, porém está resumida. O `m` minúsculo, geralmente significa **minutos**, mas quando ele vem sozinho, ele significa **dia e mês por extenso**.

E se quisermos exibir por extenso, somente o mês e o ano? Nesse caso, a letra será `Y` em maiúsculo.

```
static void Main(string[] args)
{
    Debug.WriteLine(data.ToString("D"));
    Debug.WriteLine(data.ToString("m"));
    Debug.WriteLine(data.ToString("Y"));
}
```

E o resultado será:

```
setembro de 2017
```

Esses formatos são úteis, pois eles são compactos, ou seja, não precisamos fica lembrando de colocar uma série de código e de pontos para formatar uma data.

Há um outro formato mais completo, que inclui data e hora. Com o `G` em maiúsculo, temos a data geral: "Dia, mês, ano, hora, minuto, segundo".

```
static void Main(string[] args)
{
    Debug.WriteLine(data.ToString("D"));
    Debug.WriteLine(data.ToString("m"));
    Debug.WriteLine(data.ToString("Y"));

    Debug.WriteLine(data.ToString("G"));
}
```

Vamos tirar os segundos:

```
static void Main(string[] args)
{
    Debug.WriteLine(data.ToString("D"));
    Debug.WriteLine(data.ToString("m"));
    Debug.WriteLine(data.ToString("Y"));

    Debug.WriteLine(data.ToString("G"));
    Debug.WriteLine(data.ToString("g"));
}
```

E assim, temos esse novo formato sem os segundos. Agora, veremos um formato bastante interessante. Imagine que precisamos armazenar a data como **string** no banco de dados, porém, depois será necessário retornar a string para o **dateTime** sem acontecer nenhum erro de conversão.

Existe um formato específico que o C# oferece, conhecido como o formato de **ida e volta**, ou *Round-Trip*. A letra que se refere a ele é `O` em maiúsculo.

```
static void Main(string[] args)
{
    Debug.WriteLine(data.ToString("D"));
    Debug.WriteLine(data.ToString("m"));
    Debug.WriteLine(data.ToString("Y"));

    Debug.WriteLine(data.ToString("G"));
    Debug.WriteLine(data.ToString("g"));

    Debug.WriteLine(data.ToString("O"));
}
```

Apareceu esse resultado pra nós:

```
2017-09-23T13:14:15.9870000
```

Esse é um formato ideal para você fazer a conversão de volta de **string** para **dateTime**, sem ter nenhuma perda de conversão.

Podemos lembrar que a letra "O", tem o formato de "vai e volta" e com isso podemos associar a esse formato. Vamos ver se conseguimos transformar essa string em dateTime.

```
static void Main(string[] args)
{
    Debug.WriteLine(data.ToString("D"));
    Debug.WriteLine(data.ToString("m"));
    Debug.WriteLine(data.ToString("Y"));

    Debug.WriteLine(data.ToString("G"));
    Debug.WriteLine(data.ToString("g"));

    Debug.WriteLine(data.ToString("O"));
    Debug.WriteLine(DateTime.Parse(data.ToString("O")).ToString("dd/MM/yyyy HH:mm:ss.fff"));
}
```

Vamos rodar a aplicação pra ver se ele converte novamente.

```
2017-09-23T13:14:15.9870000
23/09/2017 13:14:15.987
```

Muito bem! Conseguimos provar que o *Round-Trip* consegue realizar a conversão ter perdas. E se quisermos mostrar somente a hora e o minuto?

O formato é `t` em minúsculo, como vocês podem imaginar.

```
static void Main(string[] args)
{
    Debug.WriteLine(data.ToString("D"));
    Debug.WriteLine(data.ToString("m"));
    Debug.WriteLine(data.ToString("Y"));

    Debug.WriteLine(data.ToString("G"));
    Debug.WriteLine(data.ToString("g"));

    Debug.WriteLine(data.ToString("O"));
    Debug.WriteLine(DateTime.Parse(data.ToString("O")).ToString("dd/MM/yyyy HH:mm:ss.fff"));

    Debug.WriteLine(data.ToString("t"));
}
```

Então, apareceu `13:14` ! E se optarmos para que ele exiba hora, minuto, e segundo? Colocamos o "t" em maiúsculo.

```
static void Main(string[] args)
{
    Debug.WriteLine(data.ToString("D"));
    Debug.WriteLine(data.ToString("m"));
    Debug.WriteLine(data.ToString("Y"));
```

```
        Debug.WriteLine(data.ToString("G"));
        Debug.WriteLine(data.ToString("g"));

        Debug.WriteLine(data.ToString("O"));
        Debug.WriteLine(DateTime.Parse(data.ToString("O")).ToString("dd/MM/yyyy HH:mm:ss.fff"));

        Debug.WriteLine(data.ToString("t"));
        Debug.WriteLine(data.ToString("T"));
    }
```

Vimos como trabalhar com esses formatos compactos, e como eles podem ajudar no nosso dia a dia.