07

Mão na Massa - LINQ: Outros Operadores

Nesse capítulo vamos explorar alguns operadores do LINQ!

Imagine que você tenha uma sequência de nomes de meses:

```
string[] meses = { "janeiro", "fevereiro", "março", "abril", "maio", "junho", "julho", "agosto"
```

Nossa tarefa agora é listar os meses do primeiro trimestre. Como podemos fazer isso? Obviamente podemos usar um laço `for`, com índices de 0 a 2, e imprimir somente os meses com esses índices. Mas tem um jeito melhor.

Para obter uma quantidade N de elemntos a partir do início de uma sequência, podemos usar o método LINQ chamadao `Take()`:

```
Console.WriteLine("Pegar o primeiro trimestre");
Imprimir(meses.Take(3));
```

RESULTADO:

```
Pegar o primeiro trimestre
janeiro
fevereiro
março
```

E se quisermos pegar todos os meses do ano, exceto os do primeiro trimestre? Nesse caso, temos que "pular" 3 meses e pegar o restante. O verbo "pular" em inglês é "skip", e assim você tem que usar o método `Skip()`:

```
Console.WriteLine("Pegar os meses depois do primeiro trimestre");
Imprimir(meses.Skip(3));
```

RESULTADO:

```
Pegar os meses depois do primeiro trimestre
abril
maio
junho
julho
agosto
setembro
outubro
novembro
dezembro
```

Mas perceba que o método `Skip` acima pulou 3 meses mas pegou TODOS os meses até o final. E se quisermos obter uma faixa de elementos da sequência, como por exemplo os meses do terceiro trimestre?

Simples: podemos combinar os métodos `Skip` e `Take` para obter o resultado desejado!

```
Console.WriteLine("Pegar os meses do terceiro trimestre");
Imprimir(meses.Skip(6).Take(3));
```

RESULTADO:

```
Pegar os meses do terceiro trimestre
julho
agosto
setembro
```

Também podemos pegar os elementos a partir do início da sequência enquanto uma condição for verdadeira. Nesse caso, podemos usar o método `TakeWhile`. Por exemplo, se quisermos pegar uma sequência de meses e continuar pegando até encontarmos um mês cujo nome comece com a letra "s":

```
Console.WriteLine("Pegar os meses até que o mês comece com a letra 's'");
Imprimir(meses.TakeWhile(m => !m.StartsWith("s")));
```

RESULTADO:

```
Pegar os meses até que o mês comece com a letra 's'
janeiro
fevereiro
março
abril
maio
junho
julho
agosto
```

E se desejarmos obter os meses a partir do mês que inicia com a letra "s", podemos usar o método `SkipWhile`:

```
Console.WriteLine("Pular os meses até que o mês comece com a letra 's'");
Imprimir(meses.SkipWhile(m => !m.StartsWith("s")));
```

RESULTADO:

```
Pular os meses até que o mês comece com a letra 's'
setembro
outubro
novembro
dezembro
```

Agora vamos ver um pouco sobre operadores de conjuntos fornecidos pelo LINQ. Para isso, vamos declarar duas sequências de nomes de meses:

```
string[] seq1 = { "janeiro", "fevereiro", "março"};
string[] seq2 = { "fevereiro", "MARÇO", "abril" };
Console.WriteLine();
```

Perceba que colocamos o mês de março duas vezes. A segunda string começa com "M" maiúsculo, e fizemos isso de propósito. Vamos ver o porquê disso agora:

Vamos **concatenar** os elementos de duas sequências, utiliando o método `Concat` :

```
Console.WriteLine("concatenando duas sequências");
Imprimir(seq1.Concat(seq2));
```

RESULTADO:

```
concatenando duas sequências
janeiro
fevereiro
março
fevereiro
MARÇO
abril
```

Como vimos no resultado, o método `Concat` simplesmente faz uma "justaposição" dos elementos, sem levar em conta se o valor existe nas duas sequências! Por isso, temos uma duplicação dos meses "fevereiro" e "março".

Porém, existe um outro método para "juntar" duas sequências: o método `Union` . Vamos ver do que ele é capaz:

```
Console.WriteLine("união de duas sequências");
Imprimir(seq1.Union(seq2));
```

RESULTADO:

```
união de duas sequências
janeiro
fevereiro
março
MARÇO
abril
```

Veja que com o `Union` conseguimos eliminar a duplicação do mês "fevereiro", mas o mês de "março" ainda está duplicado. Notamos que isso acontece porque existe diferença de maiúsculas/minúsculas.

Para contornar o problema, podemos usar uma *sobrecarga* (overload) do método `Union` para que ele ignore essas diferenças de case:

```
Console.WriteLine("união de duas sequências com comparador IgnoreCase");
Imprimir(seq1.Union(seq2, StringComparer.CurrentCultureIgnoreCase));
```

RESULTADO:

```
união de duas sequências com comparador IgnoreCase
janeiro
fevereiro
março
abril
```

Agora sim, sucesso!