02

Operadores de conjuntos LINQ

Transcrição

Neste vídeo, veremos como trabalhar com conjuntos utilizando operadores Linq.

Teremos um novo projeto *console application*, onde declararemos as duas sequências a seguir:

```
namespace A4._2_OperadoresDeConjuntos
{
        class Program
        {
                static void Main(string[] args)
                {
                        string[] seq1 = { "janeiro", "fevereiro", "março" };
                        string[] seq2 = { "fevereiro", "MARÇO", "abril"};
```

Faremos operações de conjuntos, e a primeira ação será concatenar duas sequências.

Com isto, queremos que as duas sequências sirvam para fornecer os elementos para um resultado de consulta Linq, vide diagrama abaixo.

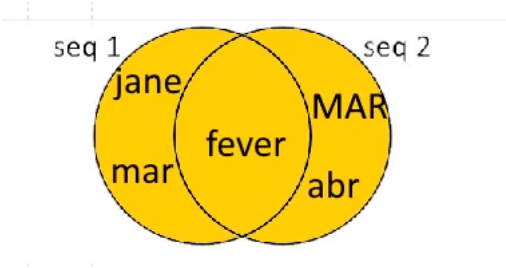
Declararemos uma consulta Linq:

```
namespace A4._2_OperadoresDeConjuntos
{
        class Program
        {
                static void Main(string[] args)
                {
                        string[] seq1 = { "janeiro", "fevereiro", "março" };
                        string[] seq2 = { "fevereiro", "MARÇO", "abril"};

                        Console.WriteLine("Conectanto duas sequências");

                        var consulta = seq1.Concat(seq2);
                        foreach (var item in consulta)
                        {
                                Console.WriteLine(item);
                        }
                        Console.WriteLine();
```

Executaremos a programação e, como é possível observar, todos os elementos das sequências foram impressos em forma de lista.

Faremos agora uma operação de conjuntos, bastante conhecida, que é a de união.

Na imagem abaixo, vemos as duas sequências, com um ponto de intersecção.



Como vimos anteriormente, os nomes foram impressos, inclusive, repetindo a palavra "fevereiro".

Com a operação de união de conjuntos, não será repetido o elemento em comum, que no caso é "fevereiro".

Representaremos isto em uma consulta Linq, da seguinte forma;

```
Console.WriteLine("União de duas sequências");
var consulta2 = seq1.Union(seq2);
foreach (var item in consulta2)
{
        Console.WriteLine(item);
}
Console.WriteLine();
```

Executando o programa, vemos que foram impressos no console os meses: janeiro, fevereiro, março, MARÇO e abril. Ou seja, não temos a repetição do item em comum.

O próximo passo será realizarmos a união de duas sequências com comprador.

Isto significa que, ainda que as palavras estejam em letras maiúsculas ou minúsculas, elas não serão repetidas, como é o caso da palavra "março", em nosso exemplo.

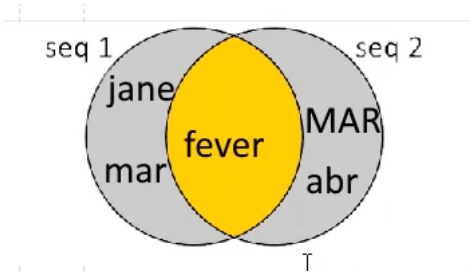
Utilizaremos um comparador, que ignore a diferença entre caixa alta e baixa.

```
Console.WriteLine("União de duas sequências com comparador");
var consulta3 = seq1.Union(seq2, StringComparer.InvariantCultureignoreCase)
foreach (var item in consulta3)
{
        Console.WriteLine(item);
}
Console.WriteLine();
```

Executando a aplicação, podemos observar a união das duas sequências com comparador. Estarão impressos os meses de janeiro, fevereiro, março e abril. Não temos mais a repetição do elemento comum (fevereiro), nem daquele que era igual, salvo pela letra maiúscula e minúscula (março).

Em seguida, faremos a intersecção entre duas sequências.

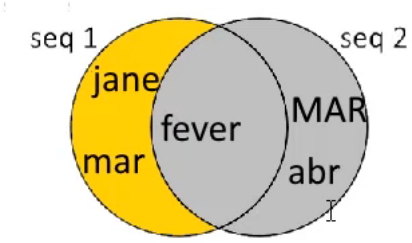
Como podemos ver no diagrama abaixo, esta operação compreende os elementos que são comuns às duas sequências.




```
Console.WriteLine("Intersecção de duas sequências");
var consulta4 = seq1.Intersect(seq2);
foreach (var item in consulta4)
{
        Console.WriteLine(item);
}
Console.WriteLine();
```

Executando a aplicação, vemos que foi impressa somente a palavra "fevereiro", que é o único elemento em comum entre as duas sequências.

Por fim, trabalharemos com o operador "exceto".



Como podemos observar no diagrama acima, ele fará com que sejam isolados todos os elementos presentes na primeira sequência, e que não estão na segunda.

```
Console.WriteLine("Exceto: elementos da seq1 que não estão em seq2");
var consulta5 = seq1.Except(seq2);
foreach (var item in consulta5)
{
        Console.WriteLine(item);
}
Console.WriteLine();
```

Executaremos a aplicação e observaremos que serão impressos, somente, aqueles elementos pertencentes à sequência 1, e que não estão na sequência 2.

Com isso, vimos quais são os operadores Linq para consultas de conjuntos.