10

Conclusão

Transcrição

Faremos, neste vídeo, um rápido resumo do conteúdo que foi ministrado neste curso.

Começamos pela coleção ordenada, ou *sorted list*, que, apesar do nome, trata-se de um dicionário.

Como podemos observar, ela é implementada com uma interface `IDictionary`. Além disso, internamente, ela funciona com duas listas, por isso o nome. Ela trabalha com uma lista de chaves e outra de valores.

Em seguida, aprendemos sobre as coleções ordenadas, ou *sorted dictionary*.

Um tipo de dicionário, no qual as chaves são organizadas por meio de uma árvore binária, e que, fora este aspecto tem funcionamento idêntico ao do *sorted list*.

O *sorted dictionary* é otimizado para que seja possível inserir e remover um grande número de elementos, em uma coleção.

Vimos, em seguida, as coleções ordenadas, ou *sorted set*.

Trata-se de um conjunto e, portanto, é possível realizar operações desse tipo em cima de uma coleção ordenada. Bem como foi possível com o *hash set*, na parte 1 do curso.

Aprendemos, ainda, a trabalhar com matrizes retangulares e multidimensionais.

A matriz retangular, como vimos, nos permitiu criar uma tabela de copas do mundo, onde uma das dimensões era o ano da referida copa, e a outra, as posições dos colocados naquele campeonato.

Foi possível organizar, em forma de tabela, as informações dentro de um array.

Vimos também como podemos trabalhar com uma matriz denteada, ou *jagged array*, que é um array de arrays. Um array que contém arrays que são, por exemplo, famílias.

No exemplo estudado, vimos que cada família continha um número diferente de elementos, ou membros.

Posteriormente, fizemos algumas consultas em cima das coleções com as quais havíamos trabalhado, utilizando a tecnologia Linq - consulta integrada à linguagem.

Vimos que, para realizar uma consulta, implementamos uma variável `IEnumerable`, de strings, e utilizamos esta consulta para podermos, por exemplo, filtrar elementos com o método `Where`, passando uma expressão lambda com a condição que desejamos filtrar.

Além disso, aprendemos a ordenar utilizando o `OrderBy`, para podermos colocar nosso resultado na ordem desejada.

Também aprendemos a transformar elementos de nossa coleção, utilizando o operador `Select`.

Em seguida, vimos como podemos obter faixas de valores dentro de coleções, utilizando o Linq, por meio dos operadores `Take` e `Skip`, que podem pular ou obter uma quantidade definida de elementos.

Conseguimos também utilizar os métodos `TakeWhile` e `SkipWhile`, para obter faixas de valores sequenciais, que correspondem a uma condição.

Aprendemos ainda a trabalhar com o Linq, para realizar operações matemáticas de conjuntos.

Por exemplo, concatenação, união e intersecção.

Passando também pelos problemas oriundos da conversão de coleções, como por exemplo, de uma lista de strings para uma de objetos.

Também convertemos um array de strings em um e objetos, e vimos os problemas que isto gera, uma vez que a covariância (característica do array) é quebrada.

Ainda, convertemos uma lista de strings em um `IEnumerable` de objetos, utilizando a covariância da interface.

Por fim, aprendemos o funcionamento interno da instrução `foreach`. Como que ela difere de funcionamento em uma lista e um array.

Vimos que o .NET Framework protege a coleção que está sendo varrida, durante o laço `foreach`, para impedir eventuais erros que comprometam a aplicação.