Microsoft .NET

Residência em Software

Tuplas, Lambda e, LINQ (e mais!)

Prof. Hélder Almeida

INSTITUIÇÃO EXECUTORA
cepedi UESC

COORDENADORA
MCTI FUTURO Softex

APOIO
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

Residência em Software

Tuplas

As tuplas no C# são uma estrutura de dados do tipo de referência que podem armazenar valores de diferentes tipos.

A principal vantagem de uma tupla é permitir o armazenamento de múltiplos tipos, como por exemplo uma string e um int ao mesmo tempo.

O recurso de tuplas fornece sintaxe concisa para agrupar vários elementos de dados em uma estrutura de dados leve.

Muito utilizado para múltiplos retornos de métodos e coleções chave-valor.

Mais informações em: https://learn.microsoft.com/pt-br/dotnet/csharp/language-reference/builtin-types/value-tuples

Residência em Software

Exemplo

```
#region Tuples Examples
var tuple1 = (10, 20);
Console.WriteLine($"Tuple 1: {tuple1.Item1}, {tuple1.Item2}");
//Tuple 1: 10, 20

var tuple2 = (x: 5, y: 50);
Console.WriteLine($"Tuple 2: {tuple2.x}, {tuple2.y}");
//Tuple 2: 5, 50

var tuple3 = (id: 10, name: "Helder", born: new DateTime(1987, 9, 24));
Console.WriteLine($"Tuple 3: {tuple3.id}, {tuple3.name}, {tuple3.born}");
// Tuple 3: 10, Helder, 24/09/1987 00:00:00

List<(int id, string name, DateTime born)> list = new();
list.Add(tuple3);
list.Add((11, "Nicole", new DateTime(2019, 1, 12)));
list.ForEach(x => Console.WriteLine($"Tuple 4: {x.id}, {x.name}, {x.born.ToShortDateString()}"));
// Tuple 4: 10, Helder, 24/09/1987
// Tuple 4: 11, Nicole, 12/01/2019
#endregion
```

Residência em Software

Expressão Lambda, o que é?

É uma função anônima que podemos usar para diversas funções.

Podemos escrever funções locais que podem ser passadas como argumentos ou retornadas como o valor de chamadas de função.

Expressões lambda podem ser escritas com uma única ou várias linhas. No entanto, na prática, normalmente não há mais de duas ou três.

Elas podem ser atribuídos a variáveis ou em linha.

Mais informações: https://learn.microsoft.com/pt-br/dotnet/csharp/language-reference/operators/lambda-expressions

Residência em Software

Exemplos

```
#region LambdaExpression Examples
Func<int, int, int> sum = (x, y) => x + y;
Console.WriteLine($"Sum: {sum(10, 20)}");
// Sum: 30

Action<string> greet = name =>
{
   string greeting = $"Hello {name}!";
   Console.WriteLine(greeting);
};
string person = Console.ReadLine() ?? "";
greet(person);
// Hello `person or Someone`
// ?? and ??= are null-coalescing operators,
//    which return the left-hand operand if the operand is not null;
//    otherwise, they return the right operand.

Func<string, int, string> isBiggerThan = (string s, int x) => s.Length > x ? "Yes" : "No";
var size = 5;
Console.WriteLine($"The text {person} has more than {size} chars? {isBiggerThan(person, size)}");

string[] people = { "Helder", "Nicole", "Gilvana" };
char letter = 'N';
Console.WriteLine($"People with name started with '{letter}': {string.Join(", ", people.Where(x => x.StartsWith(letter)))}");

#endregion
```

Residência em Software

LINQ

- LINQ (Language-Integrated Query), é um mecanismo/ferramenta para consultas de dados em .NET, compatível com diferentes fontes de dados

- Amplamente utilizado no dia a dia, pois amplia, consideravelmente, a capacidade de coleções como List.

Residência em Software

LINQ: Principais Métodos

- Any
- Single, SingleOrDefault
- First, FirstOrDefault
- OrderBy, OrderByDescending
- Where
- Select, SelectMany
- Sum, Min, Max, Count

Residência em Software

Exemplos

```
List<int> list = new() { 1, 2, 3, 4, 5 };
var squaredList = list.Select(x => x * x);
Console.WriteLine($"Original List: {string.Join(", ", list)}");
Console.WriteLine($"Squared  List: {string.Join(", ", squaredList)}");
// Original List: 1, 2, 3, 4, 5
// Squared  List: 1, 4, 9, 16, 25
var summedList = list.Select((x,index) => x + squaredList.ElementAt(index));
Console.WriteLine($"Summed   List: {string.Join(", ", summedList)}");
// Summed   List: 2, 6, 12, 20, 30

var listMultipleOfThree = list.Where(x => x % 3 == 0).ToList();
listMultipleOfThree.AddRange(squaredList.Where(x => x % 3 == 0).ToList());
listMultipleOfThree.AddRange(summedList.Where(x => x % 3 == 0).ToList());
Console.WriteLine($"List Multiple of Three: {string.Join(", ", listMultipleOfThree)}");
// List Multiple of Three: 3, 9, 6, 12, 30
Console.WriteLine($"List Multiple of Three: {string.Join(", ", listMultipleOfThree.Order())}");
```

Residência em Software

Exercícios

Exercício 1: Tuplas

Crie uma função que recebe um nome e uma idade como parâmetros e retorna uma tupla contendo o nome e a idade. Em seguida, chame a função com diferentes valores e exiba os resultados.

Exercício 2: Expressões Lambda

Defina uma expressão lambda que recebe dois números como parâmetros e retorna a soma dos quadrados desses números. Em seguida, chame a expressão lambda com alguns valores diferentes e exiba os resultados.

Exercício 3: LINQ com Lista

Crie uma lista de objetos simples, por exemplo, representando pessoas com propriedades como "Nome" e "Idade". Em seguida, use LINQ para filtrar a lista e obter todas as pessoas com idade superior a 30.

Exercício 4: LINQ com Array

Crie um array de números inteiros. Use LINQ para selecionar apenas os números pares e ordene-os de forma decrescente.

Exercício 5: Combinação de Tuplas, Expressões Lambda e LINQ

Crie uma lista de tuplas, onde cada tupla contém o nome de uma pessoa e a sua altura em centímetros. Utilize uma expressão lambda e LINQ para calcular a altura média das pessoas na lista.