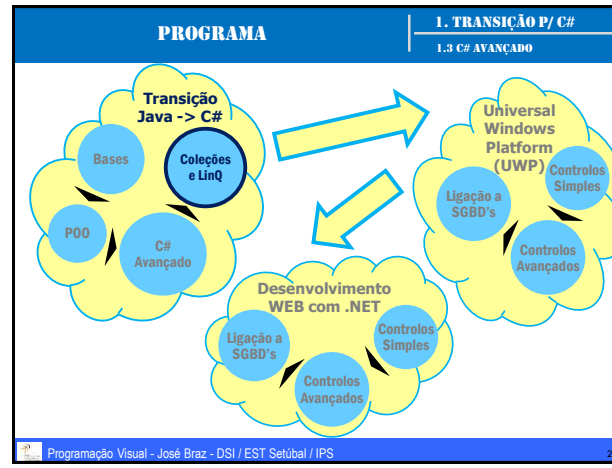
Programação Visual

Coleções 2 de 2 – LINQ

PRR REPÚBLICA PORTUGUESA Financiado pela União Europeia NextGenerationEU

1



2

C# –LINQ

- **LINQ**
  - Conceitos Básicos
  - Execução Diferida (é standard)
  - Execução Imediata (forçada)
  - Operadores Interrogativos LINQ (query operators)
  - Projecção de dados com tipos anónimos
  - Interrogações LINQ com métodos anónimos

3

Conceitos Básicos de LINQ

- O **LINQ** simplifica e uniformiza a forma de obter **informação** a partir **de um conjunto de dados**.
- O conjunto de dados pode ser qualquer coleção como por exemplo **um *array, uma lista* ou uma base de dados.**
- A linguagem LINQ está integrada no C# com uma sintaxe semelhante à da linguagem SQL (mas 'não é SQL').
- **A sintaxe mais simples** de uma instrução LINQ é:

```
from fonteDeDados in coleção
     select elementoASeleccionar
```


4

Conceitos Básicos de LINQ

```
//Dada uma coleção:
int[] nums = new int[] { 1, 2, 4, 7, 12, 14, 15, 16, 18, 21 };
```

```
//Programação em C#
List<int> res =
       new List<int>();

for (int i = 0; i < nums.Length; i++)
  if (nums[i] >= 5 && nums[i] <= 15)
    res.Add(nums[i]);

foreach( int x in res)
  Console.WriteLine("> " + x);
```

```
//Programação em C# com LINQ

var res =  from i
           in nums
           where i >= 5 &&
                 i <= 15
           select i;

foreach (int x in res)
  Console.WriteLine("> " + x);
```

res não é uma lista, é uma query

5

LINQ – execução diferida

- A query LINQ é executada no momento em que é invocada e assim o resultado traz sempre os valores atualizados.

```
List<Aluno> als = new List<Aluno> {
    new Aluno() {Nome = "Jose Antunes", Numero = 2345},
    new Aluno() {Nome = "Ana Costa", Numero = 4567},
    new Aluno() {Nome = "Manuel Jose", Numero = 6789}
};
Console.WriteLine("\n\nALUNOS com o nome 'Jose':");
var res = from a in als where a.Nome.Contains("Jose") select a.Nome;

foreach (var a in res)
 Console.WriteLine("> " + a);

Console.WriteLine("\n Jose Pinto adicionado");
als.Add(new Aluno() { Nome = "Jose Pinto", Numero = 9012 });
Console.WriteLine("\n ALUNOS com o nome 'Jose': \n");
foreach (var a in res)
  Console.WriteLine("> " + a);
```

A listagem feita aqui já mostra o aluno "Jose Pinto" introduzido imediatamente antes da invocação da query res.

6

LINQ – execução imediata

- É possível executar imediatamente uma expressão em LINQ aplicando um dos métodos definidos para o tipo **Enumerable,** como por exemplo:
  - **ToArray<T>()**, **ToList<T>()**, **ToDictionary<TSource,TKey>()**).
- Neste caso obtemos uma imagem dos dados no momento em que a query é definida (*snapshot*).

```
// expressão LINQ com execução imediata
List<String> joses =
      (from a
      in als
      where a.Nome.Contains("Jose")
      select a.Nome).ToList();

Console.WriteLine(
      "\nALUNOS com o nome 'Jose' (2):");

foreach (String n in joses)
       Console.WriteLine("> " + n);
```

Chamou-se o método **ToList()** que faz a **execução imediata** da *query* e converte o resultado para uma lista de *strings*.

Nota: na realidade estamos a invocar o método **ToList<String>()** com o tipo parametrizador omitido porque o compilador consegue determiná-lo.

7

LINQ - Operadores

- O **LINQ** utiliza na sua sintaxe **operadores** como por exemplo:
  - **from** especifica uma fonte de dados e **in** dá-lhe o contexto.
  - **select** define uma sequência de elementos a partir dos dados,
  - **where** filtra os dados com uma expressão booleana,
  - **orderby** ordenação os dados por um campo da fonte de dados
  - **ascending** e **descending** que permitem (em conjunto com o **orderby**) ordenar os resultados de forma ascendente ou descendente.
- Para além destes operadores existem vários outro tais como **join**, **on**, **equals**, **into**, **group**, **by**, etc.
- Aconselha-se a consulta:
  - https://msdn.microsoft.com/en-us/library/bb310804.aspx
  - https://msdn.microsoft.com/en-us/library/bb397927.aspx
  - https://msdn.microsoft.com/en-us/library/bb397676.aspx

8

LINQ - Operadores

- Para além dos operadores integrados na linguagem existem operadores sobre a forma de métodos de extensão genéricos que se aplicam diretamente aos resultados da *query* como, por exemplo:
  - **ToList < T > ( )** converte o resultado numa lista de elementos,
  - **ToArray< T > ( )** converte o resultado num *array* de elementos,
  - **Reverse< T > ( )** inverte a ordem dos elementos, etc.
- Além dos mencionados encontram-se alguns que fazem operações sobre conjuntos de elementos
  - **Distinct < T > ( )**
  - **Union< T > ( )**
  - **Intersect< T > ( )**
- ou outros que agregam os resultados
  - **Count< T > ( )**,
  - **Sum< T > ( )**, **Min< T > ( )**, **Max< T > ( )**, etc.).
- Recomenda-se a análise dos vários operadores disponíveis na documentação da linguagem C#.

9

LINQ – Projecção de Dados

- É possível obter como resultado de uma *query* LINQ um conjunto de dados de um tipo anónimo, subconjunto da coleção inquirida, que podem ser úteis quando se pretende analisar apenas uma parte da coleção.

```
// Projecção de dados com variáveis anónimas
List < Disciplina > disc = new List<Disciplina>();

// Adição de várias disciplinas e alunos...
// Código omitido …

var res3 = from d
           in disc
           where d.Nome.Contains("Programacao")
           select new { Nome = d.Nome, NumeroAlunos = d.TotalAlunos
};

Console.WriteLine("\nDISCIPLINAS: \n");
foreach (var r in res3)
  Console.WriteLine("> " + r.Nome + " com " + r.NumeroAlunos);
```

Foi criado um tipo anónimo com os campos **Nome** e **NumeroAlunos** que contém apenas o nome e o total de alunos na disciplina.

cria uma nova tabela, uma projeção da anterior

10

C# - Funções Anónimas

- Uma Função Anónima é uma instrução ou expressão "inline" que pode ser usada sempre que seja esperado um tipo delegate.
  - Pode ser usada para inicializar um determinado delegate
  - Ou ser passada como parâmetro de um método em lugar de um determinado delegate.
- Das funções anónimas interessam-nos as
  - Expressões Lambda

11

C# - Expressões Lambda

- Em C# uma expressão lambda é uma função anónima que pode ser usada para criar delegados.
- Para definir expressões lambda:
  - 1. Especifica-se a lista de parâmetros
  - 2. Seguida do simbolo =>
  - 3. Seguido do código a executar

```
Action<Decimal> percentMargin50LE =
    (price)
    =>
    {
      decimal profit = 0.5M * price;
      Console.WriteLine
         ("price with Profit: " +
         (profit+price));
    };

// executa a expressão Lambda
percentMargin50LE (100);
```


12

LINQ – com métodos de extensão

- É possível não utilizar a sintaxe integrada do LINQ definida para a linguagem C#. Neste caso aplicam-se directamente os métodos de extensão sobre as fontes de dados.
- Muitos dos métodos de extensão têm como argumento referências de métodos que são habitualmente chamados para cada elemento do conjunto de dados.
- Nos exemplos mostra-se a sintaxe LINQ com operadores e a mesma operação usando os métodos de extensão. Usam-se expressões lambda para passar os métodos como argumentos

```
var res4 = from a
  in als
  where a.Nome.Contains("Jose")
  select a.Nome;
```

```
var res4 =
   als.Where
   (a => a.Nome.Contains("Jose")).Select
   (a => a.Nome);
```

```
var res5 = from a
   in als
   orderby a.Nome select a;
```

```
var res5 =
   als.OrderBy
   (a => a.Nome).Select (a => a);
```

```
var res6 = from a
   in als
   where a.Numero>5000
   select a.Nome;
```

```
var res6 =
   als.Where
   (a => a.Numero>5000).Select
   (a => a.Nome);
```


13

C# - Como estudar esta matéria:

- 1. Abra o Visual Studio e abra estes slides
- 2. Abra o ficheiro A11_LINQ_Materiais.txt
- 3. Crie um projeto denominado A12_LINQ no VS
- 4.0 i = 4;
- 4.1 Enquanto i<14
- 5. i ++;
  - 5.1. Leia o slide [i] !
  - 5.2. Copie o código do ficheiro A11_LINQ.txt correspondente ao slide [i] para o método main
  - 5.3. Copie as classes necessárias para dentro do seu projeto (e apenas as necessárias para correr o código do slide [i]) Nas classes importadas comente o código desnecessário.
  - 5.4. Leia os comentários e compreenda o que se pretende ilustrar
  - 5.5. Corra o código e tente interpretar o que se está a passar
  - 5.6. Se tiver dúvidas … p e r g u n t e … please!!!
- 6 . Volte ao ponto 4.1 …

14