01

Consultando Collections

Transcrição

Começando deste ponto? Você pode fazer o download (https://github.com/marcelooliveira/csharp-collections-2/archive/19b64ef37c7a6364da186c7026339baa303cad82.zip) completo do projeto do capítulo anterior e continuar seus estudos a partir deste capítulo.

Neste curso, estudamos formas de criar novas coleções, com diferentes necessidades de aplicação.

Veremos um problema que acontece bastante, na prática.

Há diversos tipos de coleções que podem ser implementadas, e um erro comum é fazer o tipo errado de consulta ao seu acervo.

Neste exemplo, veremos como podemos obter os nomes dos meses do ano, somente daqueles que tem 31 dias, com os nomes em letras maiúsculas e em ordem alfabética.

Abaixo temos uma lista dos meses, com a quantidade de dias em cada mês.

```
static void Main(string[] args)

//PROBLEMA: obter os nomes dos meses do ano, que tem 31 dias, em maiúsculo e em ordem alfabética

//Janeiro 31
//Fevereiro 28
//Março 31
//Abril 30
//Maio 31
//Junho 30
//Julho 31
//Agosto 31
//Setembro 30
//Outubro 31
//Novembro 30
//Dezembro 31
```

Para podermos armazenar estas informações, criaremos uma classe, denominada `class Mes` .

Criaremos uma propriedade, somente leitura, digitando `propg` e pressionando a tecla "Tab" duas vezes.

Inseriremos primeiro uma com o nome dos meses, e, em seguida, com a quantidade de dias.

```
class Mes
{
        public string Nome { get; private set; }
        prop int Dias { get; private set; }
}
```

Em seguida, criaremos o construtor para esta classe. Selecionaremos as duas propriedades, e utilizaremos o atalho "Ctrl + .". Feito isso, selecionaremos a opção "*generate constructor*".

```
class Mes
{
        public Mes(string nome, int dias)
        {
                Nome = nome;
                Dias = dias;
        }

        public string Nome { get; private set; }
        public int Dias { get; private set; }
}
```

Com a classe implementada, podemos armazenar as informações dos meses.

Para isso, criaremos uma variável chamada `Mes` :

```
//PROBLEMA: obter os nomes dos meses do ano, que tem 31 dias, em maiúsculo e em ordem alfabética

List<Mes> meses = new List<Mes>
{

}

//Janeiro 31
//Fevereiro 28
//Março 31
//Abril 30
//Maio 31
//Junho 30
//Julho 31
//Agosto 31
//Setembro 30
//Outubro 31
//Novembro 30
//Dezembro 31
```

Em seguida, inseriremos a lista de meses:

```
//PROBLEMA: obter os nomes dos meses do ano, que tem 31 dias, em maiúsculo e em ordem alfabética

List<Mes> meses = new List<Mes>
{
        new Mes("Janeiro", 31),
        new Mes("Janeiro", 31),
        new Mes("Fevereiro", 28),
        new Mes("Março", 31),
        new Mes("Abril", 30),
        new Mes("Maio", 31),
        new Mes("Junho", 30),
```

```
        new Mes("Julho", 31),
        new Mes("Agosto", 31),
        new Mes("Setembro", 30),
        new Mes("Outubro", 31),
        new Mes("Novembro", 30),
        new Mes("Dezembro", 31)
};
```

Dica: para preencher os novos meses de uma só vez, utilizaremos o atalho "Alt", selecionando o espaço anterior ao nome de cada um, e em seguida pressionaremos as teclas "Ctrl + V". O mesmo processo pode ser repetido para os números de dias.

Imprimiremos esta lista, criando um laço `foreach`:

```
//PROBLEMA: obter os nomes dos meses do ano, que tem 31 dias, em maiúsculo e em ordem alfabética

List<Mes> meses = new List<Mes>
{
        new Mes("Janeiro", 31),
        new Mes("Janeiro", 31),
        new Mes("Fevereiro", 28),
        new Mes("Março", 31),
        new Mes("Abril", 30),
        new Mes("Maio", 31),
        new Mes("Junho", 30),
        new Mes("Julho", 31),
        new Mes("Agosto", 31),
        new Mes("Setembro", 30),
        new Mes("Outubro", 31),
        new Mes("Novembro", 30),
        new Mes("Dezembro", 31)
};

foreach (var mes in meses)
{
            Console.WriteLine(mes);
}
class Mes
{
        public Mes(string nome, int dias)
        {
                Nome = nome;
                Dias = dias;
        }

        public string Nome { get; private set; }
        public int Dias { get; private set; }
}
```

Executaremos o programa, utilizando o atalho "Ctrl + F5".

Como resultado, teremo o *name space*, seguido do nome da classe `Mes`.

Mas queremos imprimir, de fato, os dados dos meses. Para isso, faremos um *override* do método `to string` .

```
class Mes
{
        public Mes(string nome, int dias)
        {
                Nome = nome;
                Dias = dias;
        }

        public string Nome { get; private set; }
        public int Dias { get; private set; }

        public override string ToString(0
        {
                return $"{Nome} -{Dias}";
        }
}
```

Executaremos o programa e, como podemos obervar, agora temos os meses listados, com a quantidade de dias.

A seguir, implementaremos a consulta específica, àqueles que possuem 31 dias. Para isso, criaremos uma condição:

```
List<Mes> meses = new List<Mes>
{
        new Mes("Janeiro", 31),
        new Mes("Janeiro", 31),
        new Mes("Fevereiro", 28),
        new Mes("Março", 31),
        new Mes("Abril", 30),
        new Mes("Maio", 31),
        new Mes("Junho", 30),
        new Mes("Julho", 31),
        new Mes("Agosto", 31),
        new Mes("Setembro", 30),
        new Mes("Outubro", 31),
        new Mes("Novembro", 30),
        new Mes("Dezembro", 31)
};

foreach (var mes in meses)
{
        if (mes.Dias == 31)
        {
            Console.WriteLine(mes);
        }
}
```

Novamente, executaremos a aplicação.

Vemos que a lista conta agora, somente, com os meses com duração de 31 dias.

O próximo passo será fazer com que sejam impressos os nomes em caixa alta. Utilizaremos o método `ToUpper` :

```
foreach (var mes in meses)
{
        if (mes.Dias == 31)
        {
            Console.WriteLine(mes.Nome.ToUpper());
        }
}
```

Executando mais uma vez a aplicação, vemos que agora todos os nomes estão sendo exibidos em caixa alta.

Por fim, faremos com que os nomes sejam impressos em ordem alfabética.

Como vimos, a classe `List` utiliza o método `Sort`, para ordenação.

```
meses.Sort();
foreach (var mes in meses)
{
        if (mes.Dias == 31)
        {
            Console.WriteLine(mes.Nome.ToUpper());
        }
}
```

Executaremos o programa. Surgirá uma caixa de diálogo, informando o seguinte erro: "falha ao comparar dois elementos da matriz (...) Pelo menos um objeto deve implementar IComparable".

Isso porque estamos trabalhando com um item que é uma instância da classe `Mes`. Ela, por sua vez, não implementa a interface `IComparable`, sendo que isso é necessário para utilizarmos o método `Sort`.

Implementaremos a interface, com um método de comparação. Para isto, utilizaremos, após `IComparable`, o atalho "Ctrl + .", onde selecionaremos a opção *implement interface*.

```
class Mes : IComparable
{
        public Mes(string nome, int dias)
        {
                Nome = nome;
                Dias = dias;
        }

        public string Nome { get; private set; }'
        public int Dias { get; private set; }

        public int CompareTo(object obj)
        {
                throw new NotImplementedException();
        }

        public override string ToString(0
        {
                return $"{Nome} -{Dias}";
        }
}
```

Com isso, teremos o método `CompareTo`. O parâmetro `obj` é aquele que está sendo comparado com a nova instância, que se chamará `outro`.

Para fazer a conversão do objeto para a classe, utilizaremos o seguinte comando: `obj as Mes`.

```
meses.Sort();
foreach (var mes in meses)
{
        if (mes.Dias == 31)
        {
            Console.WriteLine(mes.Nome.ToUpper());
        }
}

class Mes : IComparable
{
        public Mes(string nome, int dias)
        {
                Nome = nome;
                Dias = dias;
        }

        public string Nome { get; private set; }'
        public int Dias { get; private set; }

        public int CompareTo(object obj)
        {
                Mes outro = obj as Mes;

                return this.Nome.CompareTo(outro.Nome);
        }

        public override string ToString(0
        {
                return $"{Nome} -{Dias}";
        }
}
```

Feito isso, executaremos mais uma vez a aplicação.

Vemos que temos listados somente os meses com 31 dias, escritos em caixa alta, e em ordem alfabética.

Entretanto, esta consulta ainda apresenta alguns problemas:

```
meses.Sort();
foreach (var mes in meses)
{
        if (mes.Dias == 31)
        {
                Console.WriteLine(mes.Nome.ToUpper());
        }
```

Ao observarmos este trecho de código, vemos que ele não nos passa exatamente a sua função. Temos primeiro uma ordenação, depois um laço `foreach `', em seguida um `if e`, no meio, temos um `ToUpper` , que transforma as letras em maiúsculas.

Isto dificulta identificarmos, visualmente, o que está acontecendo.

Isto é ruim porque a leitura é muito importante no desenvolvimento, se não conseguimos transmitir o que estamos fazendo para outro programador, a manutenção do código será dificultada. Daí a importância de termos um código que expresse sua intenção, o que não é o caso aqui.

Além disso, temos a função `meses.Sort` , que modifica a lista original. Isto não é ideal, porque se utilizarmos `meses` mais adiante, teríamos problemas, já que a lista seria alterada.

A ideia da consulta é bastante simples mas, com as ferramentas que temos, a consulta não ficou ideal. No próximo vídeo, aprenderemos a utilizar o Linq, que são consultas integradas a linguagem.