Aula 08

Prof: Henrique Augusto Maltauro

Codificar Back-end de Aplicações Web

C#

- Tipo de valor anulável

No C#, como na maior parte das linguagens de programação, temos o conceito de valores nulos, no C# representado pela palavra-chave null, onde as nossas variáveis até tem um espaço separado na memória do computador para elas, porém elas possuem valores nulos.

Ou seja, é como se aquele espaço da memória, que está separado para aquela variável, estivesse vazio.

C#

- Tipo de valor anulável

Os tipos de dados do C# não permitem, por padrão, serem definidos com valores nulos.

Ou seja, a gente não consegue definir uma variável do tipo int como null, a gente não consegue definir uma variável do tipo bool como null, e assim por diante.

C#

- Tipo de valor anulável

Mas existem situações em que se é necessário dar um valor nulo para as nossas variáveis.

Nesses casos, fazemos uma pequena modificação na declaração das nossas variáveis com um símbolo de interrogação ?.

Dessa forma, as variáveis agora podem receber um valor null.

C#

- Tipo de valor anulável

Não existe segredo, para tornar o tipo de valor como anulável, basta adicionar o símbolo de interrogação no final do tipo de valor:

```
string (Não permite valores nulos)
string? (Permite valores nulos)
```

C#

- Tipo de valor anulável

```
int? inteiroNulo = null;
string? textoNulo = null;
bool? boleanoNulo = null;
Pessoa? pessoaNula = null;
```

Trabalhando com Datas

C#

- DateTime

No C#, temos o DateTime para trabalharmos com datas. Porém o DateTime não é simplesmente um tipo de valor, ele é um pouco mais complexo do que isso.

Por definição, o DateTime é uma estrutura de tipo de valor, pois ele é composto de outros tipos de valor, como int, long e double.

C#

- DateTime

O valor do DateTime no C# pode ir da exata meia noite (00:00:00) do dia 01 do mês 01 do ano 0001 (01/01/0001), até as 23 horas, 59 minutos e 59 segundos (23:59:59) do dia 31 do mês 12 do ano 9999 (31/12/9999).

C#

- DateTime (Métodos Construtores)

A estrutura do DateTime é similar a uma classe, por isso ele possui métodos construtores que nos permitem criar os nossos valores de data.

O DateTime possui 11 métodos construtores diferentes, mas inicialmente vamos nos atentar a apenas dois deles.

C#

- DateTime (Métodos Construtores)

Primeiramente, podemos criar um valor de data básico, apenas com valores de ano, mês e dia.

```
DateTime(int year, int month, int day)
```

C#

- DateTime (Métodos Construtores)

```
// (Ano, Mês, Dia)
DateTime data = new DateTime(2023, 03, 20);
```

C#

- DateTime (Métodos Construtores)

Depois, podemos criar um valor de data mais completo, com valores de ano, mês, dia, hora, minuto e segundo.

```
DateTime(int year, int month,
         int day, int hour,
         int minute, int second)
```

C#

- DateTime (Métodos Construtores)

```
// (Ano, Mês, Dia, Hora, Minuto, Segundo)
DateTime data = new DateTime(2023, 03, 20, 19, 50, 47);
```

C#

- DateTime (Atributos)

Assim como uma classe, a estrutura do DateTime possui alguns atributos, que nos permite acessar alguns dados específicos dos valores de data.

Nesse momento, vamos nos preocupar com sete desses atributos, todos eles bem auto explicativos.

C#

- DateTime (Atributos: Now)

O atributo Now, em português Agora, nos permite acessar o valor daquele exato momento que o código estiver sendo executado.

Este atributo não depende da existência de uma variável para ser acessado.

C#

- DateTime (Atributos: Now)

```
DateTime data = DateTime.Now;
```

C#

- DateTime (Atributos: Year)

O atributo Year, em português Ano, nos permite acessar o valor do ano de uma variável do tipo DateTime.

C#

- DateTime (Atributos: Year)

```
DateTime data = DateTime.Now;
int ano = data.Year;
```

C#

- DateTime (Atributos: Month)

O atributo Month, em português Mês, nos permite acessar o valor do mês de uma variável do tipo DateTime.

C#

- DateTime (Atributos: Month)

```
DateTime data = DateTime.Now;
int mes = data.Month;
```

C#

- DateTime (Atributos: Day)

O atributo Day, em português Dia, nos permite acessar o valor do dia de uma variável do tipo DateTime.

C#

- DateTime (Atributos: Day)

```
DateTime data = DateTime.Now;
int dia = data.Day;
```

C#

- DateTime (Atributos: Hour)

O atributo Hour, em português Hora, nos permite acessar o valor das horas de uma variável do tipo DateTime.

C#

- DateTime (Atributos: Hour)

```
DateTime data = DateTime.Now;
int hora = data.Hour;
```

C#

- DateTime (Atributos: Minute)

O atributo Minute, em português Minuto, nos permite acessar o valor dos minutos de uma variável do tipo DateTime.

C#

- DateTime (Atributos: Minute)

```
DateTime data = DateTime.Now;
int minuto = data.Minute;
```

C#

- DateTime (Atributos: Second)

O atributo Second, em português Segundo, nos permite acessar o valor dos segundos de uma variável do tipo DateTime.

C#

- DateTime (Atributos: Second)

```
DateTime data = DateTime.Now;
int segundo = data.Second;
```

C#

- DateTime (Métodos)

Novamente, assim como uma classe, a estrutura do DateTime possui alguns métodos, que nos permite acessar alguns processos específicos com os valores de data.

Nesse momento, vamos nos preocupar com nove desses métodos.

C#

- DateTime (Métodos: Parse)

O método Parse nos permite converter uma string em DateTime.

Uma das maneiras de trabalharmos com esse método é recebendo dois parâmetros, um deles sendo uma string com a informação da data, e a outra sendo um IFormatProvider.

```
Parse(string s, IFormatProvider? provider)
```

C#

- DateTime (IFormatProvider)

No C#, o IFormatProvider é uma interface que fornece um processo de controle de formatação. Em outras palavras, ele nos permite definir uma configuração de formatação.

No caso do DateTime, vamos sempre fazer uso do CultureInfo, que vai receber um string para definir a configuração de data como brasileira.

C#

- DateTime (Métodos: Parse)

```
DateTime data = DateTime.Parse("20/03/2023", new CultureInfo("pt-BR"));
```

C#

- DateTime (Métodos: ParseExact)

O método ParseExact funciona de forma similar ao método Parse, com a diferença de que ele tem um parâmetro extra do tipo string, que nos permite informar qual é a formatação de data que o parâmetro de string de data possui.

```
ParseExact(string s, string format, IFormatProvider? provider)
```

C#

- DateTime (Métodos: ParseExact)

Utilizando nesse parâmetro extra dd para representar o valor de dias, MM para representar o valor de meses, yyyy para representar o valor de anos, HH para representar o valor de horas, mm para representar o valor de minutos e ss para representar o valor de segundos.

```
dd/MM/yyyy HH:mm:ss
```

C#

- DateTime (Métodos: ParseExact)

```
DateTime data = DateTime.Parse("20/03/2023", "dd/MM/yyyy", new CultureInfo("pt-BR"));
```

C#

- DateTime (Métodos: ToString)

O método ToString nos permite converter um DateTime em string, podendo receber um parâmetro do tipo string, que nos permite informar qual é a formatação que a data possui.

```
ToString(string? format)
```

C#

- DateTime (Métodos: ToString)

```
DateTime data = DateTime.Now;
string dataString = data.ToString();
////////////////////////////////////
DateTime data = DateTime.Now;
string dataString = data.ToString("dd/MM/yyyy");
```

C#

- DateTime (Métodos: AddYears)

O método AddYears nos permite adicionar uma quantidade de anos a um valor DateTime, recebendo essa quantidade de anos por um parâmetro do tipo int.

```
AddYears(int value)
```

C#

- DateTime (Métodos: AddYears)

```
DateTime data = DateTime.Now;
DateTime dataComAnosExtras = data.AddYears(4);
```

C#

- DateTime (Métodos: AddMonths)

O método AddMonths nos permite adicionar uma quantidade de meses a um valor DateTime, recebendo essa quantidade de meses por um parâmetro do tipo int.

```
AddMonths(int months)
```

C#

- DateTime (Métodos: AddMonths)

```
DateTime data = DateTime.Now;
DateTime dataComMesesExtras = data.AddMonths(4);
```

C#

- DateTime (Métodos: AddDays)

O método AddDays nos permite adicionar uma quantidade de dias a um valor DateTime, recebendo essa quantidade de dias por um parâmetro do tipo double.

```
AddDays(double value)
```

C#

- DateTime (Métodos: AddDays)

```
DateTime data = DateTime.Now;
DateTime dataComDiasExtras = data.AddDays(4.0);
```

C#

- DateTime (Métodos: AddHours)

O método AddHours nos permite adicionar uma quantidade de horas a um valor DateTime, recebendo essa quantidade de horas por um parâmetro do tipo double.

```
AddHours(double value)
```

C#

- DateTime (Métodos: AddHours)

```
DateTime data = DateTime.Now;
DateTime dataComHorasExtras = data.AddHours(4.0);
```

C#

- DateTime (Métodos: AddMinutes)

O método AddMinutes nos permite adicionar uma quantidade de minutos a um valor DateTime, recebendo essa quantidade de minutos por um parâmetro do tipo double.

```
AddMinutes(double value)
```

C#

- DateTime (Métodos: AddMinutes)

```
DateTime data = DateTime.Now;
DateTime dataComMinutosExtras = data.AddMinutes(4.0);
```

C#

- DateTime (Métodos: AddSeconds)

O método AddSeconds nos permite adicionar uma quantidade de segundos a um valor DateTime, recebendo essa quantidade de segundos por um parâmetro do tipo double.

```
AddSeconds(double value)
```

C#

- DateTime (Métodos: AddSeconds)

```
DateTime data = DateTime.Now;
DateTime dataComSegundosExtras = data.AddSeconds(4.0);
```