Fundamentos de Desenvolvimento com C#

Aula 04: Datas e Controle de Fluxo

Professor: Rinaldo Ferreira Junior

E-mail: rinaldo.fjunior@prof.infnet.edu.br

- **Professor:** Rinaldo Ferreira Junior
- **Graduação:** Pós-graduado em Arquitetura de Softwares
- **Atuação:** .Net | C# | SQL | NoSQL | Engenheiro de Software
- **E-mail:** rinaldo.fjunior@prof.infnet.edu.br
- **Linkedin:** https://www.linkedin.com/in/rinaldo-ferreira-junior-787326a

Aula 04: Roteiro

- Datas
- Estruturas de Controle de Fluxo

Aula 04: Datas

- No C#, datas são representadas pelo tipo de dados DateTime.
- DateTime possui duas propriedades representando faixa de datas válidas para manipulação:
  - MinValue é da data inicial da faixa -> January 1, 0001 00:00:00
  - MaxValue é a data máxima da faixa -> December 31, 9999 11:59:59 P.M.
- MinValue é o valor default de um tipo DateTime.
- DateTime está pronto para lidar com internacionalização, o que nos permite lidar com diferentes culturas e fusos horários.

Aula 04: Datas

- Objetos DateTime podem ser inicializados de diferentes formas:

```
static void Main(string[] args)
{
    DateTime defaultDate = DateTime.MinValue;
    DateTime currentDate = DateTime.Now;
    DateTime currentDateUtc = DateTime.UtcNow;
    DateTime dateConstructor = new DateTime(2025, 02, 18);
    DateTime dateConstructorNew = new(2025, 02, 10, 14, 34, 55);

    string dateSample = $@"Default: {defaultDate}
                        Corrente: {currentDate}
                        UTC: {currentDateUtc}
                        Construtor: {dateConstructor}
                        ConstrutorNew: {dateConstructorNew}";

    Console.WriteLine(dateSample.Trim());
    Console.Read();
}
```

Aula 04: Datas

- É possível formatar datas por diferentes meios:

```
static void Main(string[] args)
{
    DateTime currentDate = DateTime.Now;

    string dateSample = $@"Abreviada: {currentDate.ToString("dd/MM/yyyy")}
                        Data longa: {currentDate.ToString("D")}
                        Hora: {currentDate.ToString("T")}
                        Dia: {currentDate.ToString("dddd")}
                        Mês: {currentDate:MM}";

    Console.WriteLine(dateSample);
    Console.Read();
}
```

Aula 04: Datas

- DateTime possui métodos e propriedades para manipulação de seu valor.

```
static void Main(string[] args)
{
    DateTime start = new(2025, 02, 01);
    DateTime currentDate = DateTime.Now;
    DateTime tommorrow = currentDate.AddDays(1);
    DateTime yesterday = currentDate.AddDays(-1);
    int days = currentDate.Subtract(start).Days;

    string dateSample = $@"Hoje: {currentDate:dd/MM/yyyy}
                           Amanhã: {tommorrow:dd/MM/yyyy}
                           Ontem: {yesterday:dd/MM/yyyy}
                           Dias até: {days}";

    Console.WriteLine(dateSample.Trim());
    Console.Read();
}
```

Aula 04: Datas

- Parsing é o processo de validação de uma string em data.
- Diferentes métodos para validação de uma data
  - Parse()
  - TryParse()
  - ParseExact()
  - TryParseExact()
- Use Parse ou ParseExact quando você tiver certeza de que o valor checado é válido. Se não for, você deve tratar a exceção gerada.
- Use TryParse ou TryParseExact quando você quiser checar uma data sem gerar exceção. Teste a validade da data através do retorno dos métodos.

Aula 04: Datas

- Exemplos:

```
static void Main(string[] args)
{
    CultureInfo culture = new CultureInfo("en-US");

    DateTime dateTime1 = Convert.ToDateTime("10/22/2015 12:10:15 PM", culture);
    Console.WriteLine($"Convert: {dateTime1}");

    try
    {
        //O formato da data não é compatível com a cultura pt-BR
        DateTime dateTime2 = DateTime.Parse("10/22/2015 12:10:15 PM");
        Console.WriteLine($"Parse: {dateTime2}");
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Erro de Parse: {ex.Message}");
    }

    //O parsing checa o formato exato da data, conforme a cultura
    DateTime dateTime3 = DateTime.ParseExact("10-22-2015", "MM-dd-yyyy", culture);
    Console.WriteLine($"ParseExact: {dateTime3}");

    //O formato da data não é compatível com a cultura pt-BR
    bool isSuccess = DateTime.TryParse("10-22-2015", out DateTime dateTime4);
    Console.WriteLine($"TryParse resultou em: {isSuccess} para: {dateTime4}");

    //Aqui o formato da data é compatível com a cultura pt-BR
    bool isSuccessNew = DateTime.TryParse("22-10-2015", out DateTime dateTime5);
    Console.WriteLine($"TryParse resultou em: {isSuccessNew} para: {dateTime5}");

    CultureInfo provider = CultureInfo.InvariantCulture;
    bool isSuccessAgain = DateTime.TryParseExact("10-22-2015", "MM-dd-yyyy", provider, DateTimeStyles.None, out DateTime dateTime6);
    Console.WriteLine($"TryParse resultou em: {isSuccessAgain} para: {dateTime6}");
}
```

Aula 04: Estruturas de Controle de Fluxo

Instituto Infnet

- São estruturas que definem os fluxos que um programa deve seguir.
- Estruturas de seleção
  - if
  - switch
- Estruturas de laço
  - while
  - do while
  - for
  - foreach

Aula 04: Estrutura if

- Executa uma instrução caso uma expressão condicional retorne verdadeiro.

- Caso a condição não seja atendida, um fluxo adicional pode ser adicionado ao ramo else.

```
static void Main(string[] args)
{
    DateTime primeiro = new(DateTime.Now.Year, DateTime.Now.Month, 1);
    DateTime ultimo = primeiro.AddMonths(1).AddSeconds(-1);
    DateTime hoje = DateTime.Now;

    if (primeiro.DayOfWeek ≠ DayOfWeek.Sunday && primeiro.DayOfWeek ≠ DayOfWeek.Saturday)
    {
        Console.WriteLine("O primeiro dia do mês é um dia útil");
    }
    else
    {
        Console.WriteLine("O primeiro dia do mês NÃO é um dia útil");
    }
}
```

Aula 04: Estrutura if...else if

- Se a primeira condição não for satisfeita, outros fluxos de verificação podem ser adicionados.

```
static void Main(string[] args)
{
    DateTime primeiro = new(DateTime.Now.Year, DateTime.Now.Month, 1);
    DateTime ultimo = primeiro.AddMonths(1).AddSeconds(-1);
    DateTime hoje = DateTime.Now;

    if (hoje.Day == 1)
    {
        Console.WriteLine("Hoje é o primeiro dia do mês");
    }
    else if (hoje.Day == ultimo.Day)
    {
        Console.WriteLine("Hoje é o último dia do mês");
    }
    else
    {
        Console.WriteLine("Hoje é um dia qualquer");
    }
}
```

Aula 04: Estrutura switch

- Permite avaliar se uma resposta está entre possíveis respostas.

```
static void Main(string[] args)
{
    int cardNumber = 18;

    switch (cardNumber)
    {
        case 13:
            Console.WriteLine("King");
            break;
        case 12:
            Console.WriteLine("Queen");
            break;
        case 11:
            Console.WriteLine("Jack");
            break;
        case -1:
            goto case 12;
        default:
            Console.WriteLine("O card é: " + cardNumber);
            break;
    }
}
```

Aula 04: if...else vs switch

- Inicialmente é uma questão de legibilidade do código.
- Quando há muitas condições a serem testadas, switch é mais legível do que criar muitos ramos else...if.
- Há um ligeiro ganho de performance na utilização de switch, por quê não a checagem de dada condição, como ocorre no if...else...if.
  - Justificável no caso de micro otimizações
  - Advanced C# Tips: Beware of Micro-Optimizing at the Cost of Code Clarity

Aula 04: Estruturas de loop

- São estruturas usadas para manter a repetição de um trecho de código, até que uma certa condição seja satisfeita.
  - while
  - do while
  - for
  - foreach
- No while, dependendo do resultado da condição, o código pode nem ser executado.
- Já no do…while, é garantido que haja ao menos, uma execução do bloco de código.
- O for é semelhante ao while, testando o resultado de uma condição para executar o bloco de repetição
- O foreach é usado para percorrer os membros de uma coleção e interagir com eles.

Aula 04: Estruturas de loop (while)

- O teste da condição é realizado logo de início.

```
static void Main(string[] args)
{
    int i = 0;

    while (i < 5)
    {
        Console.WriteLine(i);
        i++;
    }

    Console.Read();
}
```

Aula 04: Estruturas de loop (do...while)

- O teste da condição é realizado ao final da primeira execução

```
static void Main(string[] args)
{
    int i = 0;
    do
    {
        Console.WriteLine(i);
        i++;
    }
    while (i < 5);

    Console.Read();
}
```

Aula 04: Estruturas de loop (for)

- A estrutura do for possui três elementos:
  - initializer
  - condition
  - Iterator

```
static void Main(string[] args)
{
    for (int i = 0; i < 3; i++)
    {
        Console.WriteLine("i: " + i);
    }
    Console.Read();
}
```

Aula 04: Estruturas de loop (foreach)

Instituto Infnet

- Percorre coleções acessando diretamente seus elementos, ao invés do índice desses elementos.

```
static void Main(string[] args)
{
    string[] names = { "zezinho", "huguinho", "luizinho" };
    foreach (string name in names)
    {
        Console.WriteLine(char.ToUpper(name[0]) + name.Substring(1));
    }
    Console.Read();
}
```