NDDigital Technologies

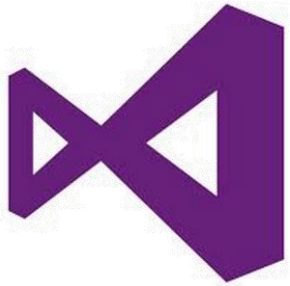
Curso preparatório Academia do programador

Instrutor: Thiago Ribeiro de Bona Sartor

NDDigital Technologies

Tratamento de erros e exceções

- Essa apresentação mostrará uma introdução sobre o tratamentos de erros e exceções.



NDDigital Technologies

Tratamento de erros e exceções

- No mundo dos frameworks e linguagens de programação, as exceções, ações que causam anomalias nas aplicações são tratadas de diversas formas. O .NET Framework elege, pelo poder e pela flexibilidade, o tratamento de exceções estruturadas. Desta forma o C# também utiliza-se deste modelo estruturado, uniforme e *type-safe.*

NDDigital Technologies

Tratamento de erros e exceções

- Quando uma exceção ocorre, um objeto herdado de *System.Exception*, é criado para representá-la. O modelo orientado á objetos permite que seja criada uma exceção definida pelo usuário que ´e herdada de *System.Exception* ou de uma outra classe de exceção pré-definida.

NDDigital Technologies

Tratamento de erros e exceções

- As exceções podem ser disparadas de duas formas: através do comando *throw*, fornecendo a instancia de uma classe herdada de *System.Exception*, ou em certas circunstancias durante o processamento dos comandos e expressões que não podem ser completadas normalmente.

NDDigital Technologies

Tratamento de erros e exceções

- Os comando em C# para utilização do tratamento de exceções estruturados são: *try* – bloco de proteção do código, *catch* - filtra e trata a exceção, *finally* - sempre executado após o disparo da exceção ou não, e *throw* - dispara uma exceção.

NDDigital Technologies

Comando throw	O comando *throw* é utilizado para disparar ou sinalizar a ocorrência de uma situação inesperada durante a execução do programa, ou seja uma exceção. O parâmetros seguido deve ser da classe *System.Exception* ou derivada.

Excessão	Descrição (disparado quando)
System.OutOfMemoryException	alocação de memória, através de new, falha.
System.StackOverflowException	quando a pilha(stack) está cheia e sobrecarregada.
System.NullReferenceException	uma referência nula(null) é utilizada indevidamente.
System.TypeInitializationException	um construtor estático dispara uma excessão.
System.InvalidCastException	uma conversão explícita falha em tempo de execução.
System.ArrayTypeMismatchException	o armazenamento dentro de um array falha.
System.IndexOutOfRangeException	o índice do array é menor que zero ou fora do limite.
System.MulticastNotSupportedException	a combinação de dois delegates não nulo falham.
System.ArithmeticException	DivideByZeroException e OverflowException. Base aritmética.
System.DivideByZeroException	ocorre uma divisão por zero.
System.OverflowException	ocorre um overflow numa operação aritmética. Checked.

NDDigital Technologies

```
// Verifica se somente uma string foi entrada
if (args.Length == 1)
    Console.WriteLine(args[0]);
else
{
    ArgumentOutOfRangeException ex;
    ex = new ArgumentOutOfRangeException("Utilize uma string somente");
    throw (ex); // Dispara a excess~ao
}
```

NDDigital Technologies

Bloco *try – catch*

- Uma ou mais instruções catch são colocadas logo abaixo do bloco *try* para interceptar uma exceção. Dentro do bloco catch ´e encontrado o código de tratamento da exceção. O tratamento da exceção trabalha de forma hierárquica, ou seja quando uma exceção ´e disparada, cada catch é verificado de acordo com a exceção e se a exceção ou derivada dela ´e encontrada o bloco será executado e os outros desprezados, por isso, na implementação é muito importante a sequência dos blocos catch. O catch também pode ser encontrado na sua forma isolada, tratando qualquer exceção não detalhada.

NDDigital Technologies

```
int iMax = 0;
Console.Write("Entre um inteiro para valor máximo, entre 0 e o máximo será sorteado: ");
try
{
    iMax = Int16.Parse(Console.ReadLine());
    Random r = new Random(); // Instância a classe Random
    int iRand = r.Next(1, iMax); // Sorteia randômincamente entre 0 e maximo
    Console.Write("O valor sorteado entre 1 e {1} é {0}", iRand, iMax);
}
catch (ArgumentException)
{
    Console.WriteLine("O não é um valor válido");
}
catch (Exception e)
{
    Console.WriteLine(e);
}
```

NDDigital Technologies

Bloco *try - finally*

- A instrução *finally* garante a execução de seu bloco, independente da exceção ocorrer no bloco *try*.
- Tradicionalmente o bloco *finally* é utilizado para liberação de recursos consumidos, por exemplo fechar um arquivo ou uma conexão.

NDDigital Technologies

```
try
{
    throw new Exception("A excessãoo. . . "); // Dispara a exceção
}
finally
{
    Console.WriteLine("O bloco finally é sempre executado. . .");
}
```

NDDigital Technologies

A classe Exception

- A forma mais comum e generalizada de disparo de uma exceção é através da classe base Exception. Ela fornece as informações necessárias para tratamento das exceções, possuindo alguns membros, métodos e propriedades, que trazem as informações necessárias decorrentes do erro. Normalmente uma instancia de classe, ou derivada, é utilizada através de um bloco catch, como vimos nos exemplos anteriores.

NDDigital Technologies

A classe Exception

- Vamos descrever alguns membros da classe Exception:
- Message: retorna uma string com o texto da mensagem de erro.
- Source: possui ou define a uma string com o texto da origem(aplica¸c˜ao ou objeto) do erro.
- HelpLink: possui uma string com o link(URN ou URL) para arquivo de ajuda.
- StackTrace: possui uma string com a sequˆencia de chamadas na stack.
- InnerException: retorna uma referˆerencia para uma excess˜ao interna.
- TargetSite: retorna o m´etodo que disparou esta excess˜ao.
- GetBaseException: retorna uma referˆerencia para uma excess˜ao interna.
- SetHelpLink: define o link(URN ou URL) para arquivo de ajuda.