Aula 15

Prof: Henrique Augusto Maltauro

Codificar Back-end de Aplicações Web

Tratamento de Exceções

Tratamento de Exceções

O tratamento de exceções é um mecanismo que permite gerenciar e controlar os erros que ocorrem durante a execução de uma aplicação.

A ideia é definir um bloco de código para a execução normal dos processos, e definir um ou mais blocos de códigos para serem executados caso ocorra um erro no bloco de código da execução normal.

Tratamento de Exceções

Podemos ainda definir um bloco de código de execução comum, que será executado independente de ter acontecido um erro ou não.

Se um erro não for tratado corretamente ele pode interromper a execução da aplicação, então o uso correto do tratamento de exceções torna a aplicação mais robusta e confiável.

Tratamento de Exceções

- C#

No C# o uso do tratamento de exceções é feito através de quatro palavras-chaves:

- try
- catch
- finally
- throw

Tratamento de Exceções

- C#: try

O try, em portuquês tente, é obrigatório e define um bloco de código para a execução normal da aplicação.

A ideia é de fato “tentar” executar o bloco de código sem que ocorra erro nenhum, e como todo bloco de código no C# ele vai ser determinado por chaves { }.

Tratamento de Exceções

- C#: try

```
try
{
    // Bloco de código que a aplicação
    // vai "tentar" executar
}
```

Tratamento de Exceções

- C#: catch

O catch, em portuquês capturar, é obrigatório e complementar ao bloco try, e define um ou mais blocos de códigos para a execução caso ocorra um erro dentro do bloco try.

A ideia é “capturar” o erro que aconteceu e executar um bloco de código de acordo com o erro capturado, e como todo bloco de código no C# ele vai ser determinado por chaves { }.

Tratamento de Exceções

- C#: catch

Ele recebe obrigatoriamente um parâmetro definido dentro de parênteses ( ), o qual irá estabelecer qual é o tipo de erro que está sendo tratado por aquele bloco de código.

Inicialmente, vamos usar o Exception que é o erro mais genérico dentro do C#, e depois vamos nos aprofundar em cada um dos erros existentes.

Tratamento de Exceções

- C#

Todos os tipos de erro do C# possuem um atributo do tipo string chamado Message, através do qual podemos obter a mensagem do erro que aconteceu durante a execução do aplicativo.

Tratamento de Exceções

- C#: catch

```
try
{
    // Bloco de código que a aplicação
    // vai "tentar" executar
}
catch (Exception ex)
{
    // Bloco de código que a aplicação
    // vai executar se acontecer
    // um erro no bloco try
}
```

Tratamento de Exceções

- C#: catch

```
try
{
    // Bloco de código que a aplicação
    // vai "tentar" executar
}
catch (Exception ex)
{
    return ex.Message;
}
```

Tratamento de Exceções

- C#: finally

O finally, em portuquês finalmente, é opcional e complementar aos blocos try e catch, e define um bloco de código para a execução independente de ter acontecido um erro ou não dentro do bloco try.

A ideia é que “finalmente” após ter executado os blocos obrigatórios, seja executado um bloco de código complementar, e como todo bloco de código no C# ele vai ser determinado por chaves { }.

Tratamento de Exceções

- C#: finally

```
try
{
    // Bloco de código que a aplicação
    // vai "tentar" executar
}
catch (Exception ex)
{
    // Bloco de código que a aplicação
    // vai executar se acontecer
    // um erro no bloco try
}
finally
{
    // Bloco de código que a aplicação
    // vai executar sempre
}
```

Tratamento de Exceções

- C#: throw

O throw, em português lançar, é uma instrução opcional, utilizada normalmente dentro do bloco try, que permite executar um erro personalizado.

A ideia é de fato “lançar” um erro para que o bloco catch “capture” ele e execute o bloco de código para o tratamento do erro.

Tratamento de Exceções

- C#: throw

O throw vai disparar um erro durante a execução da aplicação, e no C# esse erro precisa ser um tipo de objeto criado unicamente para representar um erro.

Como toda criação de objeto no C#, precisamos realizar uma instanciação do mesmo, e por hora vamos continuar a usar o tipo Exception, o qual em seguida vamos compreender melhor.

Tratamento de Exceções

- C#: throw

```
try
{
    throw new Exception();
}
catch (Exception ex)
{
    return ex.Message;
}
```

Tratamento de Exceções

- C#

Dentro do C# existem diversos tipos de exceções diferentes, todas elas podendo ser utilizadas como parâmetros do catch ou através do throw.

Não vou abordar todas elas, apenas as principais.

Tratamento de Exceções

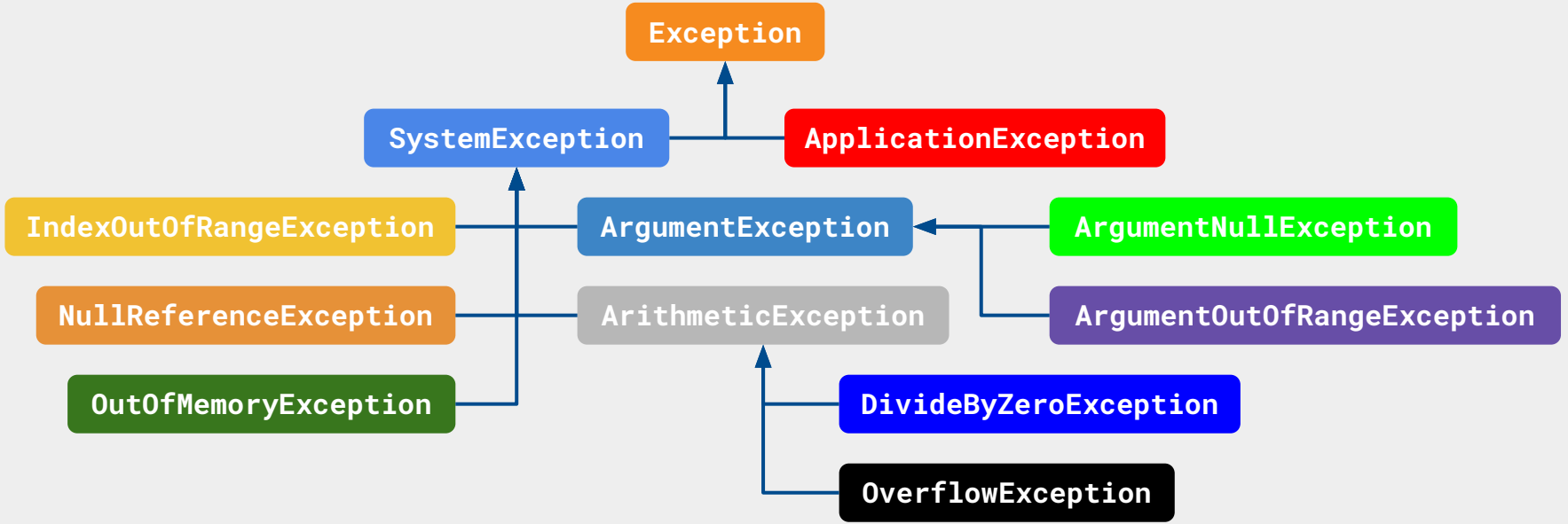
- C#

Todas as exceções são derivadas umas das outras, e sendo assim, elas possuem uma hierarquia de execução.

Essa hierarquia é executada nos blocos catch de baixo para cima.

Tratamento de Exceções

- C#



Tratamento de Exceções

- C#: Exception

O Exception é uma classe que representa qualquer tipo de exceção gerada durante a execução da aplicação.

Podemos passar uma string como parâmetro, a qual irá definir uma mensagem de erro personalizada.

Tratamento de Exceções

- C#: Exception

```
try
{
    throw new Exception();
}
catch (Exception ex)
{
    return ex.Message;
}
```

Tratamento de Exceções

- C#: Exception

```
try
{
    throw new Exception("Mensagem de erro");
}
catch (Exception ex)
{
    return ex.Message;
}
```

Tratamento de Exceções

- C#: ApplicationException

O ApplicationException é uma classe filha da classe Exception usada como base para qualquer exceção definida pela aplicação, ou seja, utilizada como herança para criar as nossas próprias exceções.

Podemos passar uma string como parâmetro, a qual irá definir uma mensagem de erro personalizada.

Tratamento de Exceções

- C#: ApplicationException

```
try
{
    throw new ApplicationException();
}
catch (ApplicationException ex)
{
    return ex.Message;
}
```

Tratamento de Exceções

- C#: ApplicationException

```
try
{
    throw new ApplicationException("Mensagem de erro");
}
catch (ApplicationException ex)
{
    return ex.Message;
}
```

Tratamento de Exceções

- C#: ApplicationException

```
public class MyCustomException : ApplicationException
{
    // Exceção personalizada
}
```

Tratamento de Exceções

- C#: SystemException

O SystemException é uma classe filha da classe Exception que funciona como base para qualquer exceção gerada pelo sistema, ou seja, qualquer exceção gerada nativamente pelo C#.

Podemos passar uma string como parâmetro, a qual irá definir uma mensagem de erro personalizada.

Tratamento de Exceções

- C#: SystemException

```
try
{
    throw new SystemException();
}
catch (SystemException ex)
{
    return ex.Message;
}
```

Tratamento de Exceções

- C#: SystemException

```
try
{
    throw new SystemException("Mensagem de erro");
}
catch (SystemException ex)
{
    return ex.Message;
}
```

Tratamento de Exceções

- C#: IndexOutOfRangeException

O IndexOutOfRangeException é uma classe filha da classe SystemException que representa qualquer exceção gerada quando ocorre uma tentativa de acessar um elemento de um vetor ou de uma lista com um índice que está fora dos limites, ou seja, um elemento que não existe no vetor ou na lista.

Podemos passar uma string como parâmetro, a qual irá definir uma mensagem de erro personalizada.

Tratamento de Exceções

- C#: IndexOutOfRangeException

```
try
{
    throw new IndexOutOfRangeException();
}
catch (IndexOutOfRangeException ex)
{
    return ex.Message;
}
```

Tratamento de Exceções

- C#: IndexOutOfRangeException

```
try
{
    throw new IndexOutOfRangeException("Mensagem de erro");
}
catch (IndexOutOfRangeException ex)
{
    return ex.Message;
}
```

Tratamento de Exceções

- C#: NullReferenceException

O NullReferenceException é uma classe filha da classe SystemException que representa qualquer exceção gerada quando ocorre uma tentativa de acessar métodos ou atributos de um objeto nulo.

Podemos passar uma string como parâmetro, a qual irá definir uma mensagem de erro personalizada.

Tratamento de Exceções

- C#: NullReferenceException

```
try
{
    throw new NullReferenceException();
}
catch (NullReferenceException ex)
{
    return ex.Message;
}
```

Tratamento de Exceções

- C#: NullReferenceException

```
try
{
    throw new NullReferenceException("Mensagem de erro");
}
catch (NullReferenceException ex)
{
    return ex.Message;
}
```

Tratamento de Exceções

- C#: OutOfMemoryException

O OutOfMemoryException é uma classe filha da classe SystemException que representa qualquer exceção gerada quando não há memória suficiente para continuar a execução da aplicação.

Podemos passar uma string como parâmetro, a qual irá definir uma mensagem de erro personalizada.

Tratamento de Exceções

- C#: OutOfMemoryException

```
try
{
    throw new OutOfMemoryException();
}
catch (OutOfMemoryException ex)
{
    return ex.Message;
}
```

Tratamento de Exceções

- C#: OutOfMemoryException

```
try
{
    throw new OutOfMemoryException("Mensagem de erro");
}
catch (OutOfMemoryException ex)
{
    return ex.Message;
}
```

Tratamento de Exceções

- C#: ArgumentException

O ArgumentException é uma classe filha da classe SystemException que representa qualquer exceção gerada quando um dos parâmetros fornecidos para um método não é válido.

Podemos passar uma string como parâmetro, a qual irá definir uma mensagem de erro personalizada.

Tratamento de Exceções

- C#: ArgumentException

```
try
{
    throw new ArgumentException();
}
catch (ArgumentException ex)
{
    return ex.Message;
}
```

Tratamento de Exceções

- C#: ArgumentException

```
try
{
    throw new ArgumentException("Mensagem de erro");
}
catch (ArgumentException ex)
{
    return ex.Message;
}
```