Bài 07

Namespace và xử lý ngoại lệ

Nội dung

- Giới thiệu về Namespace
- Đặt điểm của Namespace
- Các Namespace của hệ thống
- Các sử dụng Namespace
- Cách tạo Namespace
- Namespace lồng nhau
- Đặt bí danh cho Namespace
- Giới thiệu về Exception
- Lớp Exception
- Một số Exception của hệ thống
- Xử lý ngoại lệ
- Tạo Custom Exception

Giới thiệu Namespace

- Một namespace là một thành phần của chương trình C#
  - Giúp cho việc quản lý các class, interface và structures.
  - Namespace tránh tình trạng các tên trùng lặp giữ các class , interface, structures trong C#.
  - Bạn có thể tạo nhiều namespace trong một namespace.

Italy	USA
Rome	New York
Milan	Boston
Naples	Venice
Venice	Las Vegas
Messina	Chicago

Đặc điểm của Namespace

- Nó cung cấp một cấu trúc thứ bậc mà giúp cho việc chỉ định mối liên kết cho việc nhóm của các class.
  - Cho phép bạn thêm nhiều hơn các class, structure, enumeration, delegate và interface trong mỗi namespace được khai báo.
  - Bao gồm các class mà có tên duy nhất trong mỗi namespace.
  - Cho phép bạn sử dụng nhiều class với cùng tên bằng cách tạo chúng với các namespace khác nhau.
  - Tạo nên hệ thống cách mà các thành phần có thể tách rời nhau.

Một số Namespace trong hệ thống

- .NET Framework bao gồm rất nhiều namespace được tạo dựng sẵn và được phân cấp theo từng mục đích. Các namespace này được coi như là namespace hệ thống.
  - **System.Collections**
  - **System.Data**
  - **System.Diagnostics**
  - **System.IO**
  - **System.Net**
  - **System.Web**

Cách sử dụng Namespace

- Có 2 phương pháp sử dụng namespace

Class 1

```
using System;
...
Console.Read();
Console.Write();
```

Lập hình hiệu quả

Class 2

```
System.Console.Read();
...
System.Console.Write();
```

Lập hình không hiệu quả

Tạo Namespace 1-2

- C# cho phép bạn tạo các namespace với các tên thích hợp để quản lý các structures, class, interface, delegate, và các enumerations.
- Cú pháp:

```
namespace <NamespaceName>
{
        //Khai báo các thành phần
}
```

Tạo Namespace 2-2

Ví dụ

```
using System;
using HumanResource;//sử dụng namespace
namespace Bai10
{
    class Program
    {
        static void Main(string[] args)
        {
            Employee emp = new Employee();
            emp.Display();
        }
    }
}
//tạo namespace
namespace HumanResource
{
    //khai báo lớp
    class Employee
    {
        public void Display()
        {
            Console.WriteLine("Day la lop nhan vien");
        }
    }
}
```

Phạm vi truy xuất của Namespace

- Namespace luôn có phạm vi truy xuất là **public** bạn không thể áp dụng các từ khóa **private, protected, public, internal** cho namespace, do vậy namespace có phạm vi truy xuất không hạn chế.

Namespace

```
...
public namespace <NamespaceName> // error
{
...
}

protected namespace <NamespaceName> // error
{
...
}
...

private namespace <NamespaceName> // error
{
...
}
...

internal namespace <NamespaceName> // error
{
...
}
...
```

Sử dụng tên ngắn gọn

DEVMASTER.VN

Ví dụ

```
using System;
using HumanResource;//sử dụng namespace
namespace Bai10
{
    class Program
    {
        static void Main(string[] args)
        {
            Employee emp = new Employee(); //sử dụng tên ngắn gọn
            emp.Display();
        }
    }
}
//tạo namespace
namespace HumanResource
{
    //khai báo lớp
    class Employee
    {
        public void Display()
        {
            Console.WriteLine("Day la lop nhan vien");
        }
    }
}
```

Sử dụng tên đầy đủ

- Đây là một khai báo đầy đủ khi truy xuất vào một class nằm trong một namespace khác.

Sử dụng tên đầy đủ

Ví dụ

```
using System;
namespace Bai10
{
    class Program
    {
        static void Main(string[] args)
        {
            //sử dụng tên đầy đủ
            HumanResource.Employee emp = new HumanResource.Employee();
            emp.Display();
        }
    }
}
//tạo namespace
namespace HumanResource
{
    //khai báo lớp
    class Employee
    {
        public void Display()
        {
            Console.WriteLine("Day la lop nhan vien");
        }
    }
}
```

Namespace lồng nhau

- C# cho phép bạn tạo ra sự phân cấp của namespace bằng việc tạo các namespace lồng nhau.

Namespace lồng nhau

Ví dụ

```
//Định nghĩa các namespace lồng nhau
namespace Electronic
{
    namespace Sony
    {
        class XperiaZ
        {

        }
    }
    namespace Samsung
    {
        class Galaxy5
        {

        }
    }
    class Tivi
    {

    }
}
```

Namespace lồng nhau

```
using Electronic.Sony;
namespace Bai10
{
    class Program
    {
        static void Main(string[] args)
        {
            //cách 1 sử dụng lớp trong namespace
            Electronic.Samsung.Galaxy5 g5 = new Electronic.Samsung.Galaxy5();
            //cách 2
            XperiaZ z = new XperiaZ();
        }
    }
}
```

Đặt bí danh cho namespace 1-2

- Các alias là các tên tạm thời được coi như thực thể.
- Các alias được dùng khi có nhiều namespace lồng nhau được khai báo và bạn cần thiết dùng alias để làm ngắn gọn việc truy xuất tới các class và dễ dàng duy trì.
- Cú pháp

```
using <aliasName> = <namespaceName>;
```

Đặt bí danh cho namespace 2-2

Ví dụ

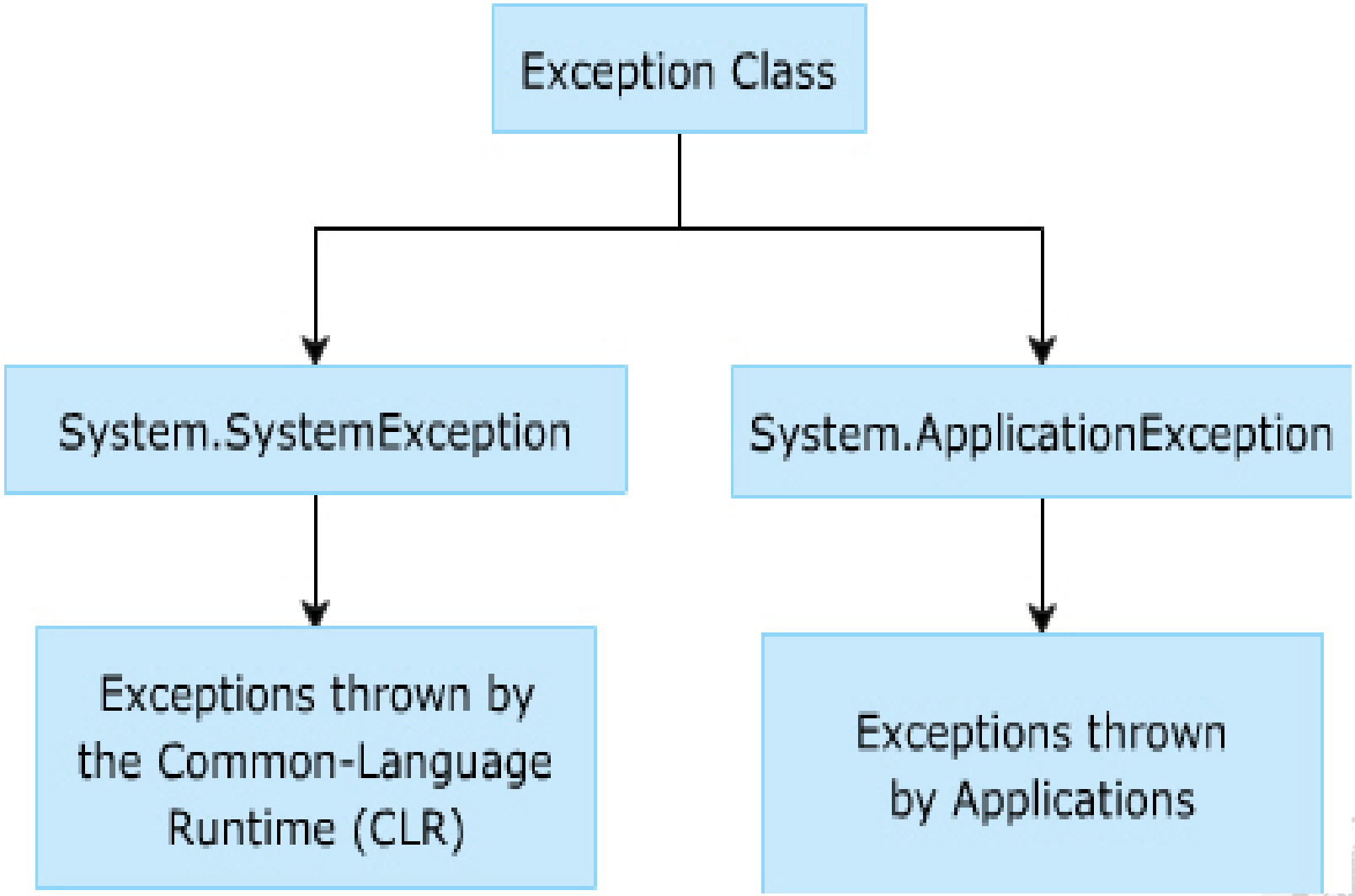
```
//Đặt bí danh
using Es = Electronic.Sony;
namespace Bai10
{
    class DemoAlias
    {
        //sử dụng lớp bên trong namespace
        Es.XperiaZ z = new Es.XperiaZ();
    }
}
namespace Electronic
{
    namespace Sony
    {
        class XperiaZ
        {

        }
    }
}
```

Xử lý ngoại lệ (Exception)

- Các exceptions là các lỗi run-time mà phá vỡ việc thực thi của một chương trình.
- Trong C# bạn có thể điều khiển sự thực thi này bằng cách sử dụng cấu trúc try...catch hoặc try...catch... finally.
- C# cho phép bạn định nghĩa các exceptions và cho phép bạn sửa đổi cách xử lý của các lỗi này.

Các ngoại lệ trong C#



Lớp Exception

- Class Exception bao gồm các phương thức public và protected có thể được kế thừa bởi các class exception khác. Thêm vào đó System. Exceptions chứa đựng các thuộc tính thuộc về tất cả các exceptions.
  - **Message:** Hiển thị một thông báo chỉ ra lý do cho exception.
  - **Source:** Cung cấp tên cho ứng dụng hoặc đối tượng trong trường hợp xảy ra exceptions.
  - **TackTrace:** Cung cấp chi tiết exception trên stack lúc exception được ném ra.
  - **InnerException:** trả về một trường hợp của exception hiện tại.

Một số lớp ngoại lệ

- Namespace System bao gồm các class exception khác nhau mà C# cung cấp.
- Các class Exception thường hay dùng là:

Tên lớp	Tên lớp
System.ArithmeticException	System.ArrayTypeMismatchException
System.DivideByZeroException	System.InvalidCastException
System.IndexOutOfRangeException	System.ArgumentNullException
System.NullReferenceException	

Điều khiển ngoại lệ

- Để điều khiển ngoại lệ C# sử dụng khối try…catch
- Cú pháp

Ví dụ

```
int a = 10, b = 0, c;
try
{
    c = a / b;
}
catch (ArithmeticException ex)
{
    Console.WriteLine("Thong bao loi:" + ex.Message);
}
```

Khối catch tổng quát

- Trong trường hợp bạn không biết rõ loại ngoại lệ nào có thể xảy ra bạn có thể sử dụng lớp Exception trong khối catch để nhận bắt bất kỳ ngoại lệ nào.

Ví dụ

```
string[] names = new string[3];
try
{
    names[3] = "Devmaster Academy";
}
catch(Exception ex)
{
    Console.WriteLine("Thông báo lỗi: "+ ex.Message);
}
```

Từ khóa “throw”

- Từ khóa “throw” dùng để tung ra một ngoại lệ, nó nhận thể hiện của lớp ngoại lệ như là 1 tham số, trong trường hợp lớp ngoại lệ tung ra không đúng, trình biên dịch sẽ báo lỗi.

Ví dụ

```
try
{
    Product p = new Product();
    p.Price = 0;
}
catch (ArgumentException ex)
{
    Console.WriteLine(ex.Message);
}
```

Từ khóa “finally”

- “finally” là khối tùy chọn trong khối try…catch, khối này luôn được thực dù ngoại lệ có xảy ra hay không, thông thường nó dùng khi thao tác với tệp tin, hoặc giao tiếp mạng.

Ví dụ

```
//Mở tệp tin
StreamWriter sw = new StreamWriter("dev.txt");
try
{
    sw.Write("Hello Devmaster Academy");//Ghi dữ liệu vào tệp tin
}
catch(Exception ex)
{
    Console.WriteLine("Thông báo lỗi:" + ex.Message);
}
finally
{
    sw.Close(); //Đóng tệp tin
}
```

Khối try…catch lồng nhau

- Một khối try…catch này lồng bên trong một khối try…catch khác và khi khối bên trong không bắt được ngoại lệ thì sẽ chuyển đến khối bên ngoài.

Khối try...catch lồng nhau

Ví dụ

```
string[] names = { "Son", "Hai", "Dong" };
int num = 0;
int result;
try
{
    Console.WriteLine("Day la khoi try catch ngoai");
    try
    {
        result = 13 / num;
    }
    catch (ArgumentException ex)
    {
        Console.WriteLine(ex.Message);
    }
    names[3] = "chung";
}
catch (IndexOutOfRangeException ex)
{
    Console.WriteLine(ex.Message);
}
```

Nhiều khối "catch"

- C# cho phép tạo nhiều khối catch để bắt các kiểu exception khác nhau được ném ra từ khối try.
- Khi khối try sinh ra exception trình biên dịch tìm khối catch tuần tự để thực hiện
  - Nếu không được trình biên dịch sẽ báo lỗi và qua khối finally nếu có.

Nhiều khối "catch"

Ví dụ

```
string[] names = { "Son", "Hai", "Dong" };
int num = 0;
int result;
try
{
    result = 13 / num;
    names[3] = "chung";
}
catch (IndexOutOfRangeException ex)
{
    Console.WriteLine(ex.Message);
}
catch (ArgumentException ex)
{
    Console.WriteLine(ex.Message);
}
```

Ngoại lệ tự định nghĩa 1-2

- Trong trường hợp những ngoại lệ của hệ thống không đáp ứng được nhu cầu bắt lỗi của bạn, bạn có thể tự định nghĩa những ngoại lệ cho chương trình của bạn
- Tuy nhiền ngoại lệ bạn tạo ra phải kế thừa từ lớp ngoại lệ chung của hệ thống là "Exception"
- Khi muốn áp dụng ngoại lệ của bạn định nghĩa, bạn dùng từ khóa "throw" để tung ra ngoại lệ

Ngoại lệ tự định nghĩa 2-2

Ví dụ

```
class Product
{
    private int price;
    public int Price
    {
        set
        {
            if (value < 0)
                //tung ngoại lệ do người dùng định nghĩa
                throw new InvalidPriceException("Gia phai >=0");
            else
                price = value;
        }
    }
}
//ngoại lệ do người dụng định nghĩa
class InvalidPriceException : Exception
{
    public InvalidPriceException(string msg)
        : base(msg)
    {
    }
}
```

Hỏi Đáp