Kivételkezelés

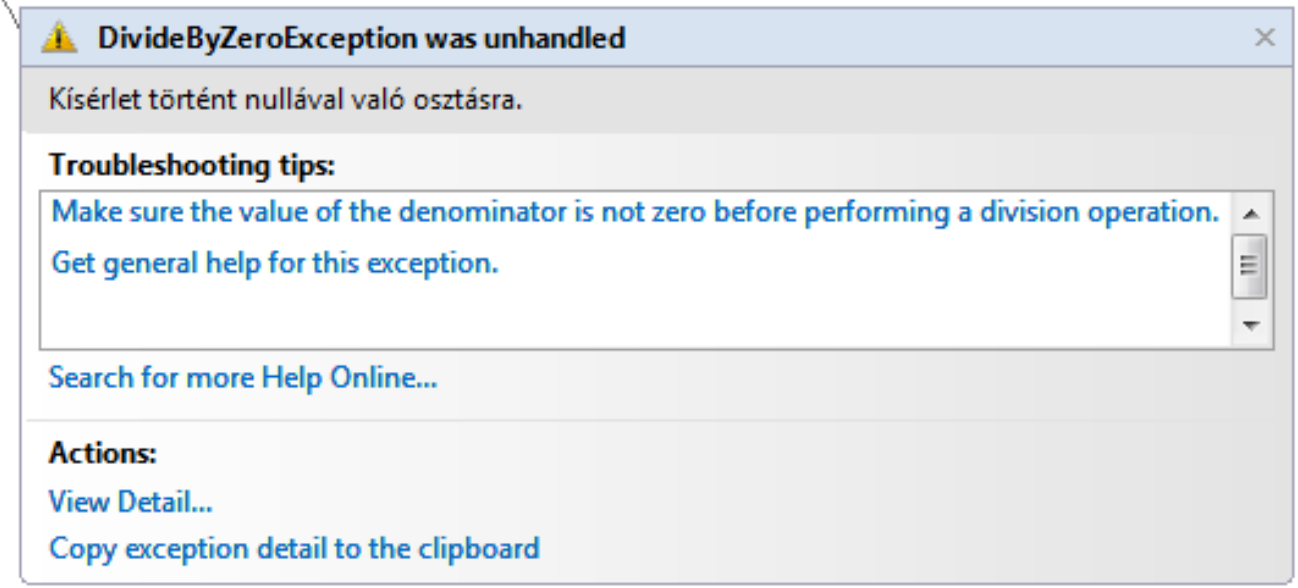
Készítette: Vastag Atila

2017

Vegyük a következő esetet:

```
class Program
{
    static void Main(string[] args)
    {
        int x = 20;
        int y = 0;
        int h = x / y;

    }
}
```

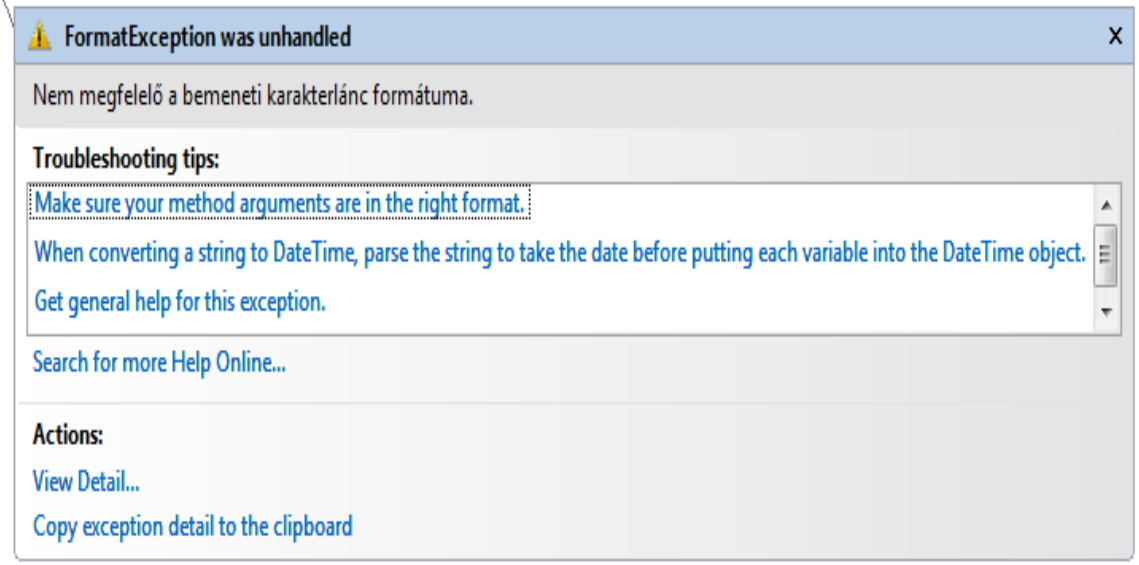


- Nullával való osztás miatt kapjuk a hibaüzenetet.

Példa II

Vegyük egy másik esetet:

```
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Kérek egy egész számot: ");
        int a = Int32.Parse(Console.ReadLine());
    }
}
```



- A hibaüzenetet azért kapjuk mert a felhasználói bevitelnél nem egész számot adtunk meg!

Kivételek

Nyílván vannak olyan esetek, amikor az alkalmazásunk, bár gond nélkül lefordul, mégsem úgy fog működni, ahogy elképzeltük. Az ilyen „abnormális” működés kezelésére találták ki a kivételkezelést. Amikor az alkalmazásunk „rossz” állapotba kerül, akkor egy ún. kivételt fog dobni.

Ilyen problémák láthatóak a :

Példa I – nullával próbáltunk osztani, ami nem lehetséges

Példa II – ahol *int* típusú változóba próbáltunk *string* típusú változót eltárolni

Ilyennel már találkoztunk a tömböknél is, amikor túlindexeltünk.

Természetesen mi azt szeretnénk, hogy valahogy kijavíthassuk ezt a hibát, ezért el fogjuk kapni a kivételt. Ehhez a művelethez három dologra van szükségünk: kijelölni azt a programrészt, ami dobhat kivételt, elkapni azt és végül kezeljük a hibát:

Kivétel keletkezésének két lehetősége van:

- Egyik lehetőség amikor a keretrendszer generálja hibát, amelyet ha nem kapunk el, akkor az operációs rendszertől kapjuk a hibaüzenetet és a program leáll
- Másik lehetőség, hogy mi is tudunk kivételt dobni, amennyiben valamilyen hibalehetőséget találunk programunk logikájában és ezt kezeljük

```
try
{
   Védett blokk, azon utasítások
    amelyek hibát okozhatnak.
}
catch (Exception ex)
{
 Itt történik a hiba fajtájától függő
    hiba kezelése. Olyan hibát
 kezelünk le, amilyet a catch után
            elkaptunk.
   throw new Exception();
}
finally
{
    Ezen utasítások mindenképp
            végrehajtódnak.
}
```

A kivétel kezeléséhez 4 új utasítást vezetünk be:

- **try**: ezt követő blokkot nevezzük védett blokknak, itt keletkezhet a hiba
- **catch**: ezzel az utasítással kaphatjuk el a védett blokk után a hibát
- **throw**: saját kivétel dobása, kivétel továbbadása
- **finally**: végzáradék, a védett blokk végén biztosan végrehajtódik

Példa I.

```
try
{
    Console.WriteLine("Kérek egy egész számot: ");
    int a = Int32.Parse(Console.ReadLine());
}
catch (Exception e)
{
    Console.WriteLine(e.Message);
}

finally
{
    Console.WriteLine("Vége");
}
```

Itt keletkezhet a hiba

Ha van hiba itt kapjuk el!

Ezzel kiíratjuk az elkapott hiba üzenetét!

Ez a blokk mindenképp lefut!

Néhány speciális hibaüzenet

- **System.ArithmeticException**
- **System.FormatException**
- **System.DivideByZeroException**
- **System.IndexOutOfRangeException**

Amennyiben ilyen speciális hiba előfordulására számítunk, úgy a catch után őt kapjuk el először és csak később az általános hibaosztályt.

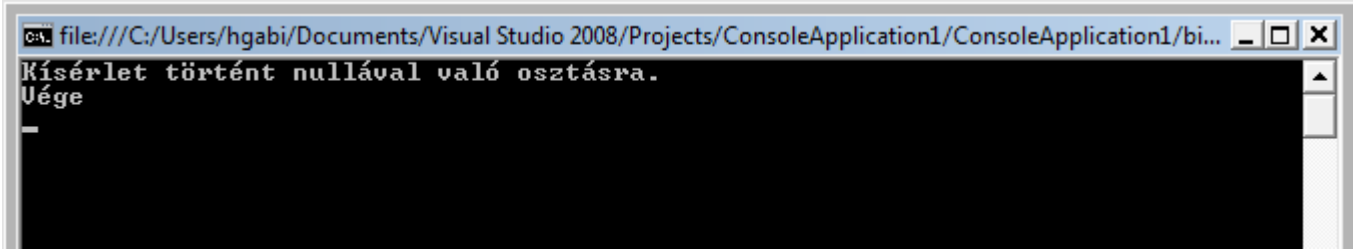
Példa III.

```
 try
{
    int x = 20;
    int y = 0;
    int h = x / y;
}
catch (DivideByZeroException hiba)
{

    Console.WriteLine(hiba.Message);
}

finally
{
    Console.WriteLine("Vége");
}
```

- Látható, hogy az előző példában már speciális hibát fogtunk meg (*DivideByZeroException*)
- Ki is írattuk, hogy hiba van, de ha még előfordulhat más hiba is a védett blokkban, akkor további **catch** blokkok alkalmazásával elkaphatjuk azokat is.
- Ügyeljünk arra, hogy ezen speciális hibák felépítése hierarchikus,először a speciális hibákat kapjuk el, majd később jöhetnek az általánosabb hibák, majd legvégül az ősosztály az Exception elkapása.
- A Base Class Library metódusaihoz a help-ben le van írva, hogy milyen típusú kivételt dobnak…



Példa III. – hogyan is kellene kivételt írni

```
try
{
    int x = 20;
    int y = 0;
    if (y == 0)
    {
        throw new DivideByZeroException("Nullával
         nem lehet osztani!!!");
    }
    else
    {
        int h = x / y;
    }
}
catch (DivideByZeroException ex)
{
    Console.WriteLine(ex.Message);
}
```

- Látható, miután megvizsgáltuk az osztó értékét és rájöttünk, hogy nulla, mi magunk indítottunk útnak egy kivételt ”Nullával nem lehet osztani!” üzenettel.
- Ezt elkapva az üzenetét kiíratva már a mi saját hibaüzenetünket láthatjuk a képernyőn
- Természetesen létrehozható saját kivétel osztály az *Exception*-ből származtatva, ezekről bővebben az MSDN-ben olvashatunk.