Manejo de excepciones

En el tutorial de hoy veremos, qué es una excepción y como trabajar con ellas.

Índice

1 - Qué es una excepción?

Una excepción es un problema o error que aparece de una manera repentina en nuestro código mientras se está ejecutando.

Debemos controlar las excepciones, ya que si no lo hacemos causara que el programa deje de trabajar, lo que implica que su funcionamiento se detiene.

La mejor forma de evitar excepciones es controlando que estas no puedan pasar. Por ejemplo, una excepción puede ocurrir cuando intentamos leer un fichero y este fichero no existe. Por lo que para evitar la excepción tendríamos que comprobar que el fichero existe, y si existe, leer, en caso contrario no leer el fichero.

Pero desafortunadamente no podemos poner un filtro o una comprobación en cada aspecto del programa. Así que lo que tenemos que hacer es un bloque `try-catch` .

2 - Bloque try-catch

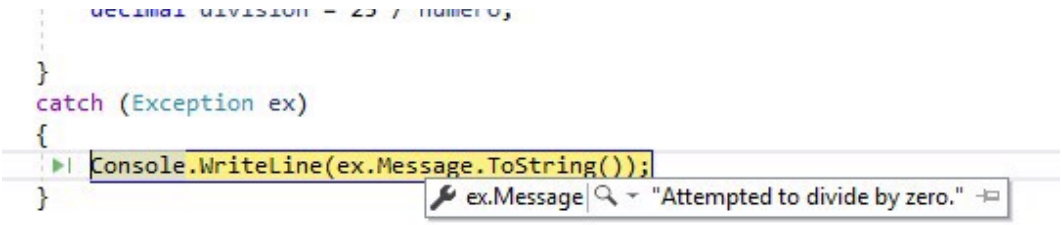
C# nos permite controlar las excepciones utilizando un bloque `try-catch` . Su funcionamiento es muy básico, ponemos código dentro del bloque `try` y si salta una excepción se ejecutará el `catch` , el cual contiene otro bloque de código. Así evitamos que el programa deje de funcionar.

Por ejemplo, no podemos dividir un número por 0 lo que hace que salte una excepción.

```
try
{
    int numero = Convert.ToInt32(Console.ReadLine());
    decimal division = 25 / numero;
}
catch (Exception ex)
{
```

```
    Console.WriteLine(ex.Message.ToString());
}
```

Co la
infc na
ocu



Otra opción muy común dentro de un catch es utilizar la clausula throw la cual enviara la excepción al bloque catch padre.

```
try
{
    decimal division = 25 / numero;
}
catch (Exception ex)
{
    throw;
}
```

Lo que quiere decir que si el método que hace saltar la excepción esta controlado en niveles superiores ejecutara el catch de los niveles superiores. En caso de no estarlo, pararíamos la ejecución del programa.

3 - Especificar una excepción

Como hemos visto esa excepción es muy concreta, ya que no se puede dividir un numero por cero.

Dentro de un `try-catch` podemos indicar tantos `catch` como queramos, siempre y cuando el tipo de dato (`Exception`) sea diferente, como vimos con los constructores, aquí también se utiliza sobrecarga de operadores.

Para nuestro ejemplo, otra posibilidad es que el numero que introducimos sea una palabra y no un número, si queremos comprobar esa excepción en concreto, debemos especificarla en el catch como vemos a continuación.

```
Console.WriteLine("Introduce un numero");

try
{
    int numero = Convert.ToInt32(Console.ReadLine());
    decimal division = 25 / numero;
}catch (FormatException ex)
{
    Console.WriteLine("El valor introducido no era un numero entero");
}
catch(DivideByZeroException ex)
{
    Console.WriteLine("No es posible dividir por 0");
}
catch (Exception ex)
{
    throw;
}
```

Como podemos observar si el valor introducido no es un número, la ejecución del programa continuara en el `catch(FormatException)`, si ese valor es 0 continuara en `catch(DivideByZeroException)` y si la excepción es cualquier otra saltara al `catch(Exception)` el cual es el bloque por defecto.

4 - Conclusión

El manejo de excepciones es algo primordial en el día a día laboral, por eso hay que tenerlo muy en cuenta cuando realizamos nuestros proyectos personales, o mientras estudiamos, ya que le dará una robustez al sistema ante fallos y problemas catastróficos.