**Try {**

using( SqlConnection conn= new SqlConnection(cadena))

{

//cod

}//close conn

**}**

**catch (Exception ex) {**

**// Handle generic ones here.**

**}**

~~Finally~~

~~{~~

~~}~~

using( SqlConnection conn= new SqlConnection(cadena))

{

//cod

}//close conn

Try {

// code here

}

catch (SqlException odbcEx) {

// Handle more specific SqlException exception here.

}

catch (Exception ex) {

// Handle generic ones here.

}

Una excepción siempre se propaga hasta que alguien la controle, ya sea que se encuentre en la misma clase u otra. Por lo tanto con estos dos ejemplos tenemos el mismo resultado:

try

{

conn.Close();

}

catch (InvalidOperationException ex)

{

Console.WriteLine(ex.GetType().FullName);

Console.WriteLine(ex.Message);

}

class TestTryCatch

{

static int GetInt(int[] array, int index)

{

try

{

return array[index];

}

catch (System.IndexOutOfRangeException e) // CS0168

{

System.Console.WriteLine(e.Message);

// Set IndexOutOfRangeException to the new exception's InnerException.

throw new System.ArgumentOutOfRangeException("index parameter is out of range.", e);

}

}

}

class CustomException : Exception

{

public CustomException(string message)

{

}

}

private static void TestThrow()

{

CustomException ex =

new CustomException("Custom exception in TestThrow()");

throw ex;

}

static void TestCatch()

{

try

{

TestThrow();

}

catch (CustomException ex)

{

System.Console.WriteLine(ex.ToString());

}

}

static void TestCatch2()

{

System.IO.StreamWriter sw = null;

try

{

sw = new System.IO.StreamWriter(@"C:\test\test.txt");

sw.WriteLine("Hello");

}

catch (System.IO.FileNotFoundException ex)

{

// Put the more specific exception first.

System.Console.WriteLine(ex.ToString());

}

catch (System.IO.IOException ex)

{

// Put the less specific exception last.

System.Console.WriteLine(ex.ToString());

}

finally

{

sw.Close();

}

System.Console.WriteLine("Done");

}

static void TestFinally()

{

System.IO.FileStream file = null;

//Change the path to something that works on your machine.

System.IO.FileInfo fileInfo = new System.IO.FileInfo(@"C:\file.txt");

try

{

file = fileInfo.OpenWrite();

file.WriteByte(0xF);

}

finally

{

// Closing the file allows you to reopen it immediately - otherwise IOException is thrown.

if (file != null)

{

file.Close();

}

}

try

{

file = fileInfo.OpenWrite();

System.Console.WriteLine("OpenWrite() succeeded");

}

catch (System.IO.IOException)

{

System.Console.WriteLine("OpenWrite() failed");

}

}

class ExceptionTest

{

static double SafeDivision(double x, double y)

{

if (y == 0)

throw new System.DivideByZeroException();

return x / y;

}

static void Main()

{

// Input for test purposes. Change the values to see

// exception handling behavior.

double a = 98, b = 0;

double result = 0;

try

{

result = SafeDivision(a, b);

Console.WriteLine("{0} divided by {1} = {2}", a, b, result);

}

catch (DivideByZeroException e)

{

Console.WriteLine("Attempted divide by zero.");

}

}

}

Ejemplo

public static Categoria ObtenerCategoriaPorId(int pIdCategoria, ref string pError)

{

using (var db = new baseContext())

{

try

{

//TODO falta implemnetar

return null;

}

catch (Exception ex)

{

pError = ex.Message;

return null;

}

}

public static void ShowSqlException(string connectionString)

{

string queryString = "EXECUTE NonExistantStoredProcedure";

StringBuilder errorMessages = new StringBuilder();

**using (SqlConnection connection = new SqlConnection(connectionString))**

**{**

**SqlCommand command = new SqlCommand(queryString, connection);**

**try**

**{**

**command.Connection.Open();**

**command.ExecuteNonQuery();**

**}**

catch (**SqlException** **ex**)

{

for (int i = 0; i < ex.Errors.Count; i++)

{

errorMessages.Append("Index #" + i + "\n" +

"Message: " + ex.Errors[i].Message + "\n" +

"LineNumber: " + ex.Errors[i].LineNumber + "\n" +

"Source: " + ex.Errors[i].Source + "\n" +

"Procedure: " + ex.Errors[i].Procedure + "\n");

}

Console.WriteLine(errorMessages.ToString());

}

**}**

}

Jerarquía de las excepciones en tiempo de ejecución

El tiempo de ejecución tiene un conjunto base de excepciones que derivan de [SystemException](https://msdn.microsoft.com/es-es/library/system.systemexception) que inicia cuando está ejecutando instrucciones individuales. En la tabla siguiente se enumeran de forma jerárquica las excepciones estándares que proporciona el tiempo de ejecución y las condiciones en que se debe crear una clase derivada.

| **Tipo de excepción** | **Tipo base** | **Descripción** | **Ejemplo** |
| --- | --- | --- | --- |
| [(Excepción)](https://msdn.microsoft.com/es-es/library/5whzhsd2) | [Objeto](https://msdn.microsoft.com/es-es/library/system.object) | Clase base de todas las excepciones. | Ninguno (utilice una clase derivada de esta excepción). |
| [SystemException](https://msdn.microsoft.com/es-es/library/system.systemexception) | [(Excepción)](https://msdn.microsoft.com/es-es/library/system.exception) | Clase base de todos los errores generados en tiempo de ejecución. | Ninguno (utilice una clase derivada de esta excepción). |
| [IndexOutOfRangeException](https://msdn.microsoft.com/es-es/library/system.indexoutofrangeexception) | [SystemException](https://msdn.microsoft.com/es-es/library/system.systemexception) | El tiempo de ejecución la genera solo cuando una matriz no está correctamente indexada. | La indexación de una matriz fuera de su intervalo válido:   var i = arr[arr.Length + 1];   Dim i = arr(arr.Length + 1) |
| [NullReferenceException](https://msdn.microsoft.com/es-es/library/system.nullreferenceexception) | [SystemException](https://msdn.microsoft.com/es-es/library/system.systemexception) | El tiempo de ejecución la genera solo cuando se hace referencia a un objeto null. | object o = null; string s = o.ToString();   Dim o As Object = Nothing Dim s As String = o.ToString() |
| [AccessViolationException](https://msdn.microsoft.com/es-es/library/system.accessviolationexception) | [SystemException](https://msdn.microsoft.com/es-es/library/system.systemexception) | El tiempo de ejecución la genera solo cuando se accede a memoria no válida. | Se produce cuando se interopera con código no administrado o código administrado no seguro y se usa un puntero no válido. |
| [InvalidOperationException](https://msdn.microsoft.com/es-es/library/system.invalidoperationexception) | [SystemException](https://msdn.microsoft.com/es-es/library/system.systemexception) | Los métodos la generan si se produce un estado no válido. | Llamar el enumerador GetNext método después de quitar un elemento de la colección subyacente. |
| [ArgumentException](https://msdn.microsoft.com/es-es/library/system.argumentexception) | [SystemException](https://msdn.microsoft.com/es-es/library/system.systemexception) | Clase base de todas las excepciones de argumento. | Ninguno (utilice una clase derivada de esta excepción). |
| [ArgumentNullException](https://msdn.microsoft.com/es-es/library/system.argumentnullexception) | [ArgumentException](https://msdn.microsoft.com/es-es/library/system.argumentexception) | Los métodos que no permiten que un argumento sea null la generan. | String s = null; int i = "Calculate".IndexOf(s);   Dim s As String = Nothing Dim i As Integer = "Calculate".IndexOf(s) |
| [ArgumentOutOfRangeException](https://msdn.microsoft.com/es-es/library/system.argumentoutofrangeexception) | [ArgumentException](https://msdn.microsoft.com/es-es/library/system.argumentexception) | Los métodos que comprueban que los argumentos se encuentran en un intervalo determinado la generan. | String s = "string"; s = s.Substring(s.Length + 1);   Dim s As String = "string" s = s.Substring(s.Length + 1) |
| [ExternalException](https://msdn.microsoft.com/es-es/library/system.runtime.interopservices.externalexception) | [SystemException](https://msdn.microsoft.com/es-es/library/system.systemexception) | Clase base de las excepciones que se producen o van dirigidas a entornos fuera del tiempo de ejecución. | Ninguno (utilice una clase derivada de esta excepción). |
| [COMException](https://msdn.microsoft.com/es-es/library/system.runtime.interopservices.comexception) | [ExternalException](https://msdn.microsoft.com/es-es/library/system.runtime.interopservices.externalexception) | Excepción que encapsula información de COM HRESULT. | Se utiliza en la interoperabilidad COM. |
| [SEHException](https://msdn.microsoft.com/es-es/library/system.runtime.interopservices.sehexception) | [ExternalException](https://msdn.microsoft.com/es-es/library/system.runtime.interopservices.externalexception) | Excepción que encapsula información de control de excepciones estructurada de Win32. | Se utiliza en interoperabilidad de código no administrado. |