**使用using或try catch來釋放非託管資源**

**何謂非託管資源:**

* **非經由 .Net 封裝定義的，稱為非託管資源Unmanaged。**
* **不受CLR(Common Language Runtime)管理的便被稱為Unmanaged非託管資源。**
* **非託管資源不被GC(Garbage Collection)所管理，使用完畢需要自行釋放。**
* **非託管資源類型必須使用IDisposable 接口的Dispose() 方法釋放。**
* **例如: Stream、與資料庫的連結、COM物件……等。**

**如何釋放非託管資源:**

顧名思義，非託管資源並不被GC所控制，須由使用者自行釋放，若未適時地將非託管資源釋放，很容易會造成資源耗盡的情況。釋放非託管資源，需要自行呼叫Dispose()方法去釋放資源，如下圖。

**public** **void** ExecuteCommand(**string** connString, **string** CommandString)

{

SqlConnection conn = **new** SqlConnection(connString);

SqlCommand cmd = **new** SqlCommand(CommandString, conn);

conn.Open();

cmd.ExecuteNonQuery();

cmd.Dispose();

conn.Dispose();

}

以上程式碼順利的話可以結束執行後，呼叫Dispose()，但當執行過程發生exception，Dispose()將無法如期被呼叫。

**使用Try catch語法釋放:**

**public** **void** ExecuteCommand(**string** connString, **string** CommandString)

{

SqlConnection conn = **null**;

SqlCommand cmd = **null**;

**try**

{

conn = **new** SqlConnection(connString);

cmd = **new** SqlCommand(CommandString, conn);

conn.Open();

cmd.ExecuteNonQuery();

}

**finally**

{

**if** (cmd != **null**)

cmd.Dispose();

**if** (conn != **null**)

conn.Dispose();

}

}

如此可以確保無論程式執行過程中有無exception，都會在finally段落中，呼叫到Dispose()。

**使用using 陳述式釋放:**

**public** **void** ExecuteCommand(**string** connString, **string** CommandString)

{

**using** (SqlConnection conn = **new** SqlConnection(connString))

{

**using** (SqlCommand cmd = **new** SqlCommand(CommandString, conn))

{

conn.Open();

cmd.ExecuteNonQuery();

}

}

}

使用 using 陳述式有一個最基本的條件，就是該物件必須有實做 IDisposable 介面，才能確保在 using 的結尾時自動執行 Dispose() 方法。該物件被建立的語法必須寫在 using 子句中。爾使用using時，相當於程式編譯時會自動變換成try finally，要特別注意並不包含catch，所以當要在using抓例外訊息時，還是區要另外寫try catch。

**參考資料:**

<http://blog.miniasp.com/post/2009/10/12/About-CSharp-using-Statement-misunderstanding-on-try-catch-finally.aspx>  
<http://sedc.pixnet.net/blog/post/14184015-c%23---%E8%B3%87%E6%BA%90%E9%87%8B%E6%94%BE%E7%9A%84%E8%A7%80%E5%BF%B5%E6%95%B4%E7%90%86>

<https://dotblogs.com.tw/larrynung/2011/03/10/21774>