# 參考資料

<https://ithelp.ithome.com.tw/articles/10193784>

# 簡介

為了要對每一筆資料做特定的處理，

LINQ會使用委派來將外面的方法帶入巡覽的時候執行，以得到期望的資料。

>委派的方式有三種 :

1. 具名函式。
2. 匿名函式。
3. Lambda運算式。

|  |
| --- |
| static string nameMethod(string output)  {  return output;  }  static void Main()  {  // 具名函式  consoleWriteLine(nameMethod, "name method");  // 匿名函式  consoleWriteLine(delegate (string output) { return output; },  "anonymous method");  // Lambda運算式  consoleWriteLine(output => output, "lambda method");  }  //public delegate TResult Func<in T, out TResult>(T arg);  private static void consoleWriteLine(Func<string, string> stringGetter, stringinput)  {  Console.WriteLine(stringGetter(input));  } |

# 具名函式

將已宣告的方法(nameMethod)指給委派。

# 匿名函式

delegate (arguments) { statements }

* delegate: 匿名函式的的保留字
* arguments: 傳入參數的宣告，可以多個參數(以,隔開)
* statements: 此函式執行的程式碼片段

詳細參考 : <https://docs.microsoft.com/zh-tw/dotnet/csharp/language-reference/language-specification/expressions#anonymous-function-expressions>

# Lambda運算式

arguments => expression | block

* arguments: 傳入參數的宣告，可以多個參數(以,隔開)。

1. 只有一個參數時可以不用括號，複數個參數都要加上括號。
2. 可以不用明確指定型別。
3. 明確指定型別時一定要加上括號。

沒有傳入參數時用空括號()表示。

* expression: 運算式，不括大括號{}，只能單行程式碼，代表回傳值。
* block: { statements }: 程式碼區塊，statement為此函式執行的程式碼片段。

## arguments

ex :

|  |
| --- |
| //Only one input parameter  x => x \* x  (x) => x \* x  //Two or more input parameters  x, y => x \* y // (X)  (x, y) => x \* y |

ex : 可以明確指定型別(一定要加上括號”()”)，不明確指定型別也可以。

|  |
| --- |
| string x => x \* x // (X)  (string x) => x \* x  x => x \* x |

ex : 沒有傳入參數時以空括號()表示

|  |
| --- |
| () => Console.WriteLine("Hello Lambda") |

## expression

只有單行程式碼時為expression，可以不用{}包住程式，程式行的最後也不用加”;”，其程式碼所代表的是回傳值。

ex :

|  |
| --- |
| x => x \* x //delegate (int x) { return x \* x; } |

使用expression格式的Lambda稱為Lambda運算式(Lambda Expressions)。

## block

一般的程式碼區塊，可以多行程式碼，每行最後要加 ”;”。

|  |
| --- |
| x => { return x \* x; } |

使用block格式的Lambda稱為Lambda陳述式(Lambda Statements)。