C#筆記

1. Yield return

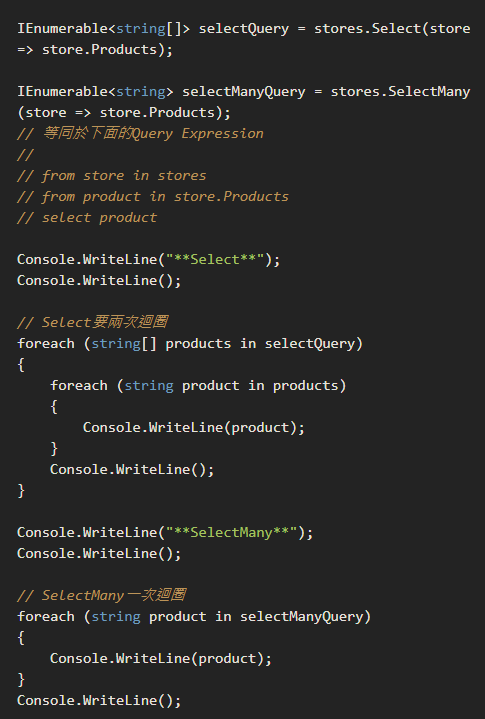
您可以使用 yield return 陳述式一次傳回一個元素。從迭代器方法傳回的序列，可以透過使用 foreach 陳述式或 LINQ 查詢來取用。 foreach 迴圈的每個反覆項目都會呼叫 Iterator 方法。 當Iterator 方法中到達 yield return 陳述式時，就會傳回 expression 並保留程式碼中目前的位置。下一次呼叫 Iterator 函式時，便會從這個位置重新開始執行。

您可以使用 yield break 陳述式結束反覆項目。

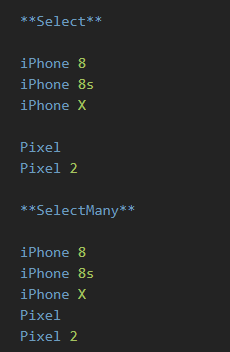
1. Select vs selectMany



好像不需要()在new store旁



Output:

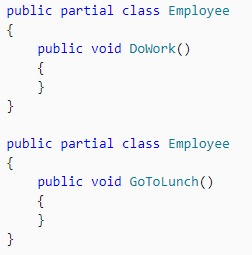


一個跟Select相似的語法-SelectMany，這個語法在處理Master/Details(主檔/明細檔)的資料時非常的有用，在沒有SelectMany前，我們處理有明細的資料都需要用複數層迴圈才能查找資料，現在我們只要用SelectMany就搞定了

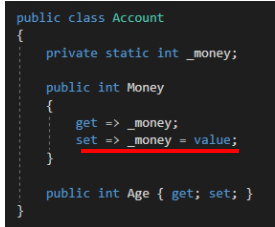
1. Partial關鍵字

有幾種情況需要分割類別定義︰

1. 處理大型專案時，將類別分散到不同的檔案可讓多位程式設計人員同時處理它。
2. 處理自動產生的來源時，程式碼可以加入類別，不必重新建立來源檔案。 Visual Studio 建立 Windows Forms、Webb 服務包裝函式程式碼等等時，會使用這種方法。 您可以建立使用這些類別的程式碼，不必修改 Visual Studio 建立的檔案。
3. 若要分割類別定義，請使用 partial 關鍵字修飾詞，如下所示︰



1. Value隱式變數: 表示傳遞進來的值



1. 關鍵字out: 引數會以傳址方式進入副程式

int initializeInMethod;

OutArgExample(out initializeInMethod);

Console.WriteLine(initializeInMethod); // value is now 44

void OutArgExample(out int number)

{

number = 44;

}

1. 方法中的擴展方法

扩展方法使您能够向现有类型“添加”方法，而无需创建新的派生类型、重新编译或以其他方式修改原始类型。扩展方法是一种特殊的静态方法，但可以像扩展类型上的实例方法一样进行调用。对于用 C# 和 Visual Basic 编写的客户端代码，调用扩展方法与调用在类型中实际定义的方法之间没有明显的差异。

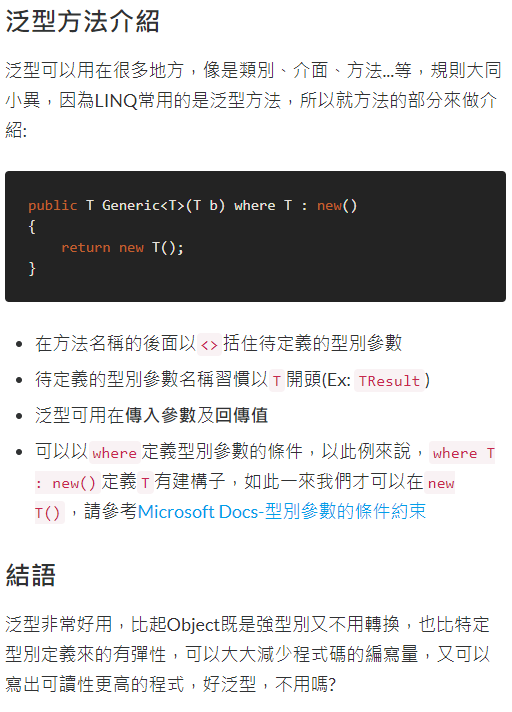
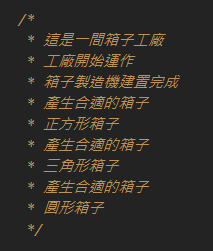
扩展方法被定义为静态方法，但它们是通过实例方法语法进行调用的。它们的第一个参数指定该方法作用于哪个类型，并且该参数以 this 修饰符为前缀。仅当您使用 using 指令将命名空间显式导入到源代码中之后，扩展方法才位于范围中。

下面的示例演示为 System.String 类定义的一个扩展方法。请注意，它是在非嵌套、非泛型静态类内部定义的：  
**namespace** **ExtensionMethods**  
**{**  
    **public** **static** **class** **MyExtensions**  
    **{**  
        **public** **static** int WordCount(**this** String str)  
        **{**  
            **return** str.Split(**new** char[] **{** ' ', '.', '?' **}**, StringSplitOptions.RemoveEmptyEntries).Length;  
        **}**  
    **}**     
**}**

　　可使用以下 using 指令将 WordCount 扩展方法放入范围中：  
　　using ExtensionMethods;  
　　而且，可以在应用程序中使用以下语法对该扩展方法进行调用：  
　　string s = "Hello Extension Methods";  
　　int i = s.WordCount();

1. 泛型用法範例 (<T>, return T())





1. Action、Fúnc、Delegate

委派的目的: 保留程式實作部分給外部程式。

1. Action: 封裝無回傳值的方法
2. Func: 封裝回傳一個值的方法
3. Func的最後一個T，永遠是返回的值，其餘的則是傳入的參數。範例: Func<T, TResult>
4. Lambda C#
5. 運算式: (input-parameters) => expression

Func<int, int> square = x => x \* x;

Console.WriteLine(square(5));

// Output:

// 25

int[] numbers = { 2, 3, 4, 5 };

var squaredNumbers = numbers.Select(x => x \* x);

Console.WriteLine(string.Join(" ", squaredNumbers));

1. 陳述式: (input-parameters) => { <sequence-of-statements> }

Action<string> greet = name =>

{

string greeting = $"Hello {name}!";

Console.WriteLine(greeting);

};

greet("World");

// Output:

// Hello World!

Action line = () => Console.WriteLine();

1. $: 字串插補

Console.WriteLine($"The low and high temperature on {weatherData.Date:MM-DD-YYYY}");

Console.WriteLine($" was {weatherData.LowTemp} and {weatherData.HighTemp}.");

// Output (similar to):

// The low and high temperature on 08-11-2020

// was 5 and 30.

1. virtual: 覆寫父類別中虛擬方法的方法需要利用 override 關鍵字，避免意外重複定義。
2. int? 表示該宣告變數可為null值
3. 匿名型別

var v = new { Amount = 108, Message = "Hello" };

// Rest the mouse pointer over v.Amount and v.Message in the following

// statement to verify that their inferred types are int and string.

Console.WriteLine(v.Amount + v.Message);

1. 只會傳回product的color與price屬性資料

var productQuery =

from prod in products

select new { prod.Color, prod.Price };

foreach (var v in productQuery)

{

Console.WriteLine("Color={0}, Price={1}", v.Color, v.Price);

}

1. 成員存取範圍

public：存取不受限制。

private：存取僅限於此類別。

protected：存取僅限於此類別或衍生自這個類別的類別。

internal：存取限於目前元件 (.exe 或 .dll) 。

protected internal：存取僅限於此類別、衍生自這個類別的類別，或相同元件中的類別。

private protected：存取權僅限於此類別或在相同元件中衍生自此類型的類別。