

Lesson 8-3: Iterate through a collection using foreach

One of the most common things you'll need to do with collections is to iterate through each item in the collection. Iterating through a collection is a little like flicking through the pages in a book.

If you knew the number of items in a collection, you could directly address them using `Collection[0]`, `Collection[1]`, etc. but most of the time you won't know in advance how many items your collection has.

In this lesson you will use a *foreach* loop to iterate through each item in a collection.

- 1 Open *My Project* from your sample files folder.
- 2 Open *CalculatorFunctions.cs*.
- 3 Add a method to total all of the numbers in a list of *int* variables.

You should be quite familiar with creating methods by now. Use the following code:

```
public static int TotalNumbers(List<int> ListToTotal)
{
}
```

```
public static int TotalNumbers(List<int> ListToTotal)
{
}
```

This method accepts a *List<int>* collection as an argument and returns an *int* value. When this method is complete it will add up the values of each element in the *ListToTotal* collection and then return the total.

This method is *static*, so you won't need to create an instance of the *CalculatorFunctions* class to call the method.

- 4 Create an *int* variable called: **Total**

To return the total as an *int* value, you'll need to create an *int* variable to return. Create one in the *TotalNumbers* method using the following code:

```
int Total = 0;
```

```
public static int TotalNumbers(List<int> ListToTotal)
{
    int Total = 0;
}
```

The *Total* variable is assigned a value of 0. You'll add the value of each element in the *ListToTotal* collection to the *Total* variable as you iterate through them.

- 5 Iterate through each item in *ListToTotal* using *foreach*.

Add the following code to the *TotalNumbers* method:

note

Using *var* iteration variables with *foreach* is bad practice

You learned about the *var* variable type in: *Lesson 5-13: Use object and var variables.*

If you use a *var* iteration variable in a *foreach* statement, it will automatically be assigned the same variable type as the items in the collection.

If, in this example, you'd used:

```
foreach (var NumberToTotal in ListToTotal)
```

...the code would have worked in exactly the same way.

It is best practice to avoid using *var* variables, as it makes the code harder to understand.

This is a common "bad" programming practice. I've included this information because you may see this practice in other people's code.

```
foreach (int NumberToTotal in ListToTotal) { }
```

```
int Total = 0;
foreach (int NumberToTotal in ListToTotal)
{
}
```

The *NumberToTotal* variable is called the *Iteration Variable*. The code will loop through all of the items in the *ListToTotal* collection. For each iteration, the value of the current item will appear in the *NumberToTotal* iteration variable.

6 Add code to sum the total in the *Total* variable.

Add the following code inside the curly brackets following the *foreach* statement:

```
Total = Total + NumberToTotal;
```

As the *foreach* statement iterates through each item in the *ListToTotal* collection, their values will be added to the *Total* variable.

7 Add code to return the total value.

At the end of the method, you'll need to return the *Total* variable using: **return Total;**

```
int Total = 0;
foreach (int NumberToTotal in ListToTotal)
{
    Total = Total + NumberToTotal;
}
return Total;
```

8 Try out the method on the *calculator.aspx* page.

1. Open the code-behind file of *calculator.aspx*.
2. Remove all code from the *ButtonCalculate2_Click* event handler.
3. Add code to the method to create a *List* of *int* variables, containing some values:

```
List<int> ListToTotal = new List<int>() { 12, 17, 135 };
```

This *List* collection contains the integer values 12, 17 and 135. Therefore its total should come to 164.


4. Add code to call the method from the *CalculatorFunctions* class and output the result:

```
int Total = CalculatorFunctions.TotalNumbers(ListToTotal);
Response.Write(Total);
```

```
protected void ButtonCalculate2_Click(object sender, EventArgs e)
{
    List<int> ListToTotal = new List<int>() { 12, 17, 135 };
    int Total = CalculatorFunctions.TotalNumbers(ListToTotal);
    Response.Write(Total);
}
```

5. View *calculator.aspx* in your browser and click the second button (named *Button*).

The number 164 is displayed, showing that your method totaled the numbers correctly!

 http://localhost:51163/calculator.aspx

164