**Introduction**

This article provides simple examples for working with [Tuples](https://msdn.microsoft.com/en-us/library/system.tuple%28v=vs.110%29.aspx?f=255&MSPPError=-2147217396). In the Microsoft documentation for the new tuple the code samples are written towards experienced developer while the novice developer will generally not grasp possibilities for using the new tuples.

**Definition of Tuple prior to Visual Studio 2017**.

A tuple is a data structure that has a specific number and sequence of elements. An example of a tuple is a data structure with three elements (known as a 3-tuple or triple) that is used to store an identifier such as a person's name in the first element, a year in the second element, and the person's income for that year in the third element. The .NET Framework directly supports tuples with one to seven elements. In addition, you can create tuples of eight or more elements by nesting tuple objects in the Rest property of a Tuple(Of T1, T2, T3, T4, T5, T6, T7, TRest) object.

Although tuples had a place in writing code the results sent back from a method returning a tuple were not readily understood as each tuple returns as (where “result” is the tuple) result.ItemN where N is a value such as first name. So if a method using a tuple returned first and last name we would have result.Item1 for first name and result.Item2 for last name.

An alternate would be to create a simple class with two properties, FirstName and LastName, create an instance of the class and return the values so we would have result.FirstName and result.LastName.

**Basic Example**

Visual Studio 2017, VB.NET 15.3 changed this by allowing a developer to create named values for tuples returning from a method call. When you instantiate the tuple, you define the number and the data type of each value (or element). For example, a 2-tuple (or pair) has two elements. For example, you have a strong typed list of Person class and want to return only first and last name you can write code as shown below.

Public Function FindPersonByIdentifierAsTuple(pIdentifier As Integer) As (FirstName As String, LastName As String)

Dim personData = People.FirstOrDefault(Function(person) person.Identifier = pIdentifier)

If personData Is Nothing Then

Return ("", "")

Else

Return (personData.FirstName, personData.LastName)

End If

End Function

Which would be called as follows

Dim identifier As Integer = 3

Dim ops = New Operations

Dim results = ops.FindPersonByIdentifierAsTuple(identifier)

If Not String.IsNullOrWhiteSpace(results.FirstName) Then

MessageBox.Show($"{results.FirstName} {results.LastName} for id of {identifier}")

Else

MessageBox.Show($"Failed to locate a person with the id of {identifier}")

End If

The quark here for some is checking to see if results.FirstName is an empty string which indicates the passed identifier was not located. Some might consider simple returning a class instance as shown below which is perfectly okay to do yet what about if the class Person has many properties and moving parts were one example might be a Entity Framework entity such as Person with Account members? In this case the returning container of the instance Person is heavy rather than light weight as in the tuple example shown above.

Public Function FindPersonByIdentifierAsPerson(pIdentifier As Integer) As Person

Return People.FirstOrDefault(Function(person) person.Identifier = pIdentifier)

End Function

**anonymous type replacement**

A perfect example for using the new tuple is writing a LINQ or Lambda query which returns an anonymous type, using named tuples a developer can return two or more values from a method as shown below where a character is passed in, a Lambda statement finds the character and returns the character, Occurrences of that character and the code for the character.

Public Function CouldHaveBeenAnonymousResults(pChar As Char) As (Item As Char, Occurrences As Integer, Code As Integer)

Dim results =

(

From T In

(

From c In "T0\*A1?0\*23aTA3 4T4\+a4 ?407#?A\*6T+".ToCharArray()

Group c By c Into Group Select New With

{

.Item = c,

.Occurrences = Group.Count,

.Code = Asc(c)

}

).ToList.OrderBy(Function(x) x.Item)

).FirstOrDefault(Function(x) x.Item = pChar)

Return (results.Item, results.Occurrences, results.Code)

End Function

**Summary**

Using the new style tuple provides additional options for returning values from a function when more than one value needs to be returned and on the caller side the members of the tuple are easily understandable and strong typed.

**Other resources**

[Microsoft documentation on tuples](https://docs.microsoft.com/en-us/dotnet/visual-basic/programming-guide/language-features/data-types/tuples).

**Source code**

In the source code there are three projects, one for demonstrating using a class instance to return data, the second for performing the same operation as the first project using named tuples while the third project demonstrates the alternate to working with anonymous types for dealing with returning information from a method. Note there are several classes that are there to show what could be returned if in the class example would be returned rather than using a light weight solution as with tuples.

Requires

From NuGet package manager in your solution, add System.ValueTuple from the “Browse tab” or from NuGet console PM>Install-Package System.ValueTuple -Version 4.5.0

<https://www.nuget.org/packages/System.ValueTuple/>

There are times when a new feature becomes available to developers that has no usage in their current solution but will try to fit the feature in even when it makes zero sense. Other new features just do not feel right and may have a code smell.

One of these are known as tuples which permit multiple values to be returned from a method call. For example, pass in a customer identifier and return first and last name. With Tuples of old first name comes back as Item1, last name comes back as item2 which is not intuitive. Other solutions would be to pass two strings ByRef or create a class with two properties for first and last name.

With the new implementation of Tuples using [NuGet ValueTuple](https://www.nuget.org/packages/System.ValueTuple/) with VB.NET 15.3 and higher a Tuple in the case above would have FirstName rather than Item1 and lastName rather than Item2. These are called named tuple.

<https://docs.microsoft.com/en-us/dotnet/visual-basic/programming-guide/language-features/data-types/tuples>