**Object and Collection Initializers (C# Programming Guide)**

Object initializers let you assign values to any accessible fields or properties of an object at creation time without having to explicitly invoke a constructor. The following example shows how to use an object initializer with a named type. Note the use of auto-implemented properties in the Test class. For more information, see [Auto-Implemented Properties (C# Programming Guide)](http://msdn.microsoft.com/en-us/library/bb384054.aspx).

C#

[[http://i.msdn.microsoft.com/Global/Images/clear.gif](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl02CSharp');)Copy Code](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl02CSharp');)

private class Cat

{

// Auto-implemented properties

public int Age { get; set; }

public string Name { get; set; }

}

static void MethodA()

{

// Object initializer

Cat cat = new Cat { Age = 10, Name = "Sylvester" };

}

http://i.msdn.microsoft.com/Global/Images/clear.gif Object Initializers with anonymous types

Although object initializers can be used in any context, they are especially useful in LINQ query expressions. Query expressions make frequent use of anonymous types, which can only be initialized with an object initializer. In the **select** clause, a query expression can transform objects of the original sequence into objects whose value and shape may differ from the original. This is very useful if you want to store only a part of the information in each object in a sequence. In the following example, assume that a product object (p) contains many fields and methods, and that you are only interested in creating a sequence of objects that contain the product name and the unit price.

C#

[[http://i.msdn.microsoft.com/Global/Images/clear.gif](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl08CSharp');)Copy Code](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl08CSharp');)

var productInfos =

from p in products

select new { p.ProductName, p.UnitPrice };

When this query is executed, the productInfos variable will contain a sequence of objects that can be accessed in a **foreach** statement as shown in this example:

[[http://i.msdn.microsoft.com/Global/Images/clear.gif](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl09other');)Copy Code](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl09other');)

foreach(var p in productInfos){...}

Each object in the new anonymous type has two public properties which receive the same names as the properties or fields in the original object. You can also rename a field when you are creating an anonymous type; the following example renames the UnitPrice field to Price.

[[http://i.msdn.microsoft.com/Global/Images/clear.gif](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl10other');)Copy Code](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl10other');)

select new {p.ProductName, Price = p.UnitPrice};

http://i.msdn.microsoft.com/Global/Images/clear.gif Object initializers with nullable types

It is a compile-time error to use a collection initializer with a nullable struct.

http://i.msdn.microsoft.com/Global/Images/clear.gif Collection Initializers

Collection initializers let you specify one or more element intializers when you initialize a collection class that implements [IEnumerable](http://msdn.microsoft.com/en-us/library/system.collections.ienumerable.aspx). The element initializers can be a simple value, an expression or an object initializer. By using a collection initializer you do not have to specify multiple calls to the **Add** method of the class in your source code; the compiler adds the calls.

The following examples shows two simple collection initializers:

[[http://i.msdn.microsoft.com/Global/Images/clear.gif](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl21other');)Copy Code](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl21other');)

List<int> digits = new List<int> { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };

List<int> digits2 = new List<int> { 0 + 1, 12 % 3, MakeInt() };

The following collection initializer uses object initializers to initialize objects of the Cat class defined in an earlier example. Note that the individual object initializers are enclosed in braces and separated by commas.

C#

[[http://i.msdn.microsoft.com/Global/Images/clear.gif](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl22CSharp');)Copy Code](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl22CSharp');)

List<Cat> cats = new List<Cat>

{

new Cat(){ Name="Sylvester", Age=8 },

new Cat(){ Name="Whiskers", Age=2},

new Cat() { Name="Sasha", Age=14}

};

You can specify [null](http://msdn.microsoft.com/en-us/library/edakx9da.aspx) as an element in a collection initializer if the collection's **Add** method allows it.

C#

[[http://i.msdn.microsoft.com/Global/Images/clear.gif](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl24CSharp');)Copy Code](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl24CSharp');)

List<Cat> moreCats = new List<Cat>

{

new Cat(){ Name="Furrytail", Age=5 },

new Cat(){ Name="Peaches", Age=4},

null

};