**Object.MemberwiseClone Method**

**.NET Framework 4.5**

[Other Versions](javascript:;)

Description: http://i.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [.NET Framework 4](http://msdn.microsoft.com/en-us/library/system.object.memberwiseclone(d=printer,v=vs.100).aspx)
* [.NET Framework 3.5](http://msdn.microsoft.com/en-us/library/system.object.memberwiseclone(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/system.object.memberwiseclone(d=printer,v=vs.85).aspx)
* [.NET Framework 2.0](http://msdn.microsoft.com/en-us/library/system.object.memberwiseclone(d=printer,v=vs.80).aspx)
* [.NET Framework 1.1](http://msdn.microsoft.com/en-us/library/system.object.memberwiseclone(d=printer,v=vs.71).aspx)
* [Silverlight](http://msdn.microsoft.com/en-us/library/system.object.memberwiseclone(d=printer,v=vs.95).aspx)

Creates a shallow copy of the current [Object](http://msdn.microsoft.com/en-us/library/system.object(v=vs.110).aspx).

**Namespace:**  [System](http://msdn.microsoft.com/en-us/library/system(v=vs.110).aspx)  
**Assembly:**  mscorlib (in mscorlib.dll)

[Syntax](javascript:void(0))

C#

[C++](http://msdn.microsoft.com/en-us/library/system.object.memberwiseclone(d=printer,v=vs.110).aspx?cs-save-lang=1&cs-lang=cpp#code-snippet-1)

[F#](http://msdn.microsoft.com/en-us/library/system.object.memberwiseclone(d=printer,v=vs.110).aspx?cs-save-lang=1&cs-lang=fsharp#code-snippet-1)

[VB](http://msdn.microsoft.com/en-us/library/system.object.memberwiseclone(d=printer,v=vs.110).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

protected Object MemberwiseClone()

**Return Value**

Type: [System.Object](http://msdn.microsoft.com/en-us/library/system.object(v=vs.110).aspx)  
A shallow copy of the current [Object](http://msdn.microsoft.com/en-us/library/system.object(v=vs.110).aspx).

[Remarks](javascript:void(0))

The MemberwiseClone method creates a shallow copy by creating a new object, and then copying the nonstatic fields of the current object to the new object. If a field is a value type, a bit-by-bit copy of the field is performed. If a field is a reference type, the reference is copied but the referred object is not; therefore, the original object and its clone refer to the same object.

For example, consider an object called X that references objects A and B. Object B, in turn, references object C. A shallow copy of X creates new object X2 that also references objects A and B. In contrast, a deep copy of X creates a new object X2 that references the new objects A2 and B2, which are copies of A and B. B2, in turn, references the new object C2, which is a copy of C. The example illustrates the difference between a shallow and a deep copy operation.

There are numerous ways to implement a deep copy operation if the shallow copy operation performed by the MemberwiseClone method does not meet your needs. These include the following:

* Call a class constructor of the object to be copied to create a second object with property values taken from the first object. This assumes that the values of an object are entirely defined by its class constructor.
* Call the MemberwiseClone method to create a shallow copy of an object, and then assign new objects whose values are the same as the original object to any properties or fields whose values are reference types. The DeepCopy method in the example illustrates this approach.
* Serialize the object to be deep copied, and then restore the serialized data to a different object variable.
* Use reflection with recursion to perform the deep copy operation.

[Examples](javascript:void(0))

The following example illustrates the MemberwiseClone method. It defines a ShallowCopy method that calls the MemberwiseClone method to perform a shallow copy operation on a Person object. It also defines a DeepCopy method that performs a deep copy operation on a Person object.

C#

[VB](http://msdn.microsoft.com/en-us/library/system.object.memberwiseclone(d=printer,v=vs.110).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-2)

using System;

public class IdInfo

{

public int IdNumber;

public IdInfo(int IdNumber)

{

this.IdNumber = IdNumber;

}

}

public class Person

{

public int Age;

public string Name;

public IdInfo IdInfo;

public Person ShallowCopy()

{

return (Person)this.MemberwiseClone();

}

public Person DeepCopy()

{

Person other = (Person) this.MemberwiseClone();

other.IdInfo = new IdInfo(this.IdInfo.IdNumber);

return other;

}

}

public class Example

{

public static void Main()

{

// Create an instance of Person and assign values to its fields.

Person p1 = new Person();

p1.Age = 42;

p1.Name = "Sam";

p1.IdInfo = new IdInfo(6565);

// Perform a shallow copy of p1 and assign it to p2.

Person p2 = (Person) p1.ShallowCopy();

// Display values of p1, p2

Console.WriteLine("Original values of p1 and p2:");

Console.WriteLine(" p1 instance values: ");

DisplayValues(p1);

Console.WriteLine(" p2 instance values:");

DisplayValues(p2);

// Change the value of p1 properties and display the values of p1 and p2.

p1.Age = 32;

p1.Name = "Frank";

p1.IdInfo.IdNumber = 7878;

Console.WriteLine("\nValues of p1 and p2 after changes to p1:");

Console.WriteLine(" p1 instance values: ");

DisplayValues(p1);

Console.WriteLine(" p2 instance values:");

DisplayValues(p2);

// Make a deep copy of p1 and assign it to p3.

Person p3 = p1.DeepCopy();

// Change the members of the p1 class to new values to show the deep copy.

p1.Name = "George";

p1.Age = 39;

p1.IdInfo.IdNumber = 8641;

Console.WriteLine("\nValues of p1 and p3 after changes to p1:");

Console.WriteLine(" p1 instance values: ");

DisplayValues(p1);

Console.WriteLine(" p3 instance values:");

DisplayValues(p3);

}

public static void DisplayValues(Person p)

{

Console.WriteLine(" Name: {0:s}, Age: {1:d}", p.Name, p.Age);

Console.WriteLine(" Value: {0:d}", p.IdInfo.IdNumber);

}

}

// The example displays the following output:

// Original values of p1 and p2:

// p1 instance values:

// Name: Sam, Age: 42

// Value: 6565

// p2 instance values:

// Name: Sam, Age: 42

// Value: 6565

//

// Values of p1 and p2 after changes to p1:

// p1 instance values:

// Name: Frank, Age: 32

// Value: 7878

// p2 instance values:

// Name: Sam, Age: 42

// Value: 7878

//

// Values of p1 and p3 after changes to p1:

// p1 instance values:

// Name: George, Age: 39

// Value: 8641

// p3 instance values:

// Name: Frank, Age: 32

// Value: 7878

In this example, the Person.IdInfo property returns an IdInfo object. As the output from the example shows, when a Person object is cloned by calling the MemberwiseClone method, the cloned Person object is an independent copy of the original object, except that they share the same Person.IdInfo object reference. As a result, modifying the clone's Person.IdInfo property changes the original object's Person.IdInfo property. On the other hand, when a deep copy operation is performed, the cloned Person object, including its Person.IdInfo property, can be modified without affecting the original object.