.NET Framework 4

**ValueType Class**

Updated: August 2011

Provides the base class for value types.

ValueType overrides the virtual methods from [Object](http://msdn.microsoft.com/en-us/library/system.object.aspx) with more appropriate implementations for value types. See also [Enum](http://msdn.microsoft.com/en-us/library/system.enum.aspx), which inherits from ValueType.

Data types are separated into value types and reference types. Value types are either stack-allocated or allocated inline in a structure. Reference types are heap-allocated. Both reference and value types are derived from the ultimate base class [Object](http://msdn.microsoft.com/en-us/library/system.object.aspx). In cases where it is necessary for a value type to behave like an object, a wrapper that makes the value type look like a reference object is allocated on the heap, and the value type's value is copied into it. The wrapper is marked so the system knows that it contains a value type. This process is known as boxing, and the reverse process is known as unboxing. Boxing and unboxing allow any type to be treated as an object.

Although ValueType is the implicit base class for value types, you cannot create a class that inherits from ValueType directly. Instead, individual compilers provide a language keyword or construct (such as struct in C# and Structure…End Structure in Visual Basic) to support the creation of value types.

Except for serving as the base class for value types in the .NET Framework, the ValueType structure is generally not used directly in code. However, it can be used as a parameter in method calls to restrict possible arguments to value types instead of all objects, or to permit a method to handle a number of different value types. The following example illustrates how ValueType prevents reference types from being passed to methods. It defines a class named Utility that contains four methods: IsNumeric, which indicates whether its argument is a number; IsInteger, which indicates whether its argument is an integer; IsFloat, which indicates whether its argument is a floating-point number; and Compare, which indicates the relationship between two numeric values. In each case, the method parameters are of type ValueType, and reference types are prevented from being passed to the methods.