**Auto-Implemented Properties (C# Programming Guide)**

In C# 3.0 and later, auto-implemented properties make property-declaration more concise when no additional logic is required in the property accessors. They also enable client code to create objects When you declare a property as shown in the following example, the compiler creates a private, anonymous backing field can only be accessed through the property's get and set accessors.

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

The following example shows a simple class that has some auto-implemented properties:

C#

[[http://i.msdn.microsoft.com/Global/Images/clear.gif](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl06CSharp');)Copy Code](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl06CSharp');)

// This class is mutable. Its data can be modified from

// outside the class.

class Customer

{

// Auto-Impl Properties for trivial get and set

public double TotalPurchases { get; set; }

public string Name { get; set; }

public int CustomerID { get; set; }

// Constructor

public Customer(double purchases, string name, int ID)

{

TotalPurchases = purchases;

Name = name;

CustomerID = ID;

}

// Methods

public string GetContactInfo() {return "ContactInfo";}

public string GetTransactionHistory() {return "History";}

// .. Additional methods, events, etc.

}

class Program

{

static void Main()

{

// Intialize a new object.

Customer cust1 = new Customer ( 4987.63, "Northwind",90108 );

//Modify a property

cust1.TotalPurchases += 499.99;

}

The class that is shown in the previous example is mutable. Client code can change the values in objects after they are created. In complex classes that contain significant behavior (methods) as well as data, it is often necessary to have public properties. However, for small classes or structs that just encapsulate a set of values (data) and have little or no behaviors, it is recommended to make the objects immutable by declaring the set accessor as [private](http://msdn.microsoft.com/en-us/library/st6sy9xe.aspx). For more information, see [How to: Implement an Immutable Class That has Auto-Implemented Properties (C# Programming Guide)](http://msdn.microsoft.com/en-us/library/bb383979.aspx).

Attributes are permitted on auto-implemented properties but obviously not on the backing fields since those are not accessible from your source code. If you must use an attribute on the backing field of a property, just create a regular property.