Interfaces (Guía de programación de C#)

**Visual Studio 2013**

[Otras versiones](javascript:;)

http://i3.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

Este tema aún no ha recibido ninguna valoración - [Valorar este tema](http://msdn.microsoft.com/es-es/library/ms173156.aspx#feedback)

Una interfaz contiene las definiciones de un grupo de funciones relacionadas que una[clase](http://msdn.microsoft.com/es-es/library/0b0thckt.aspx) o [struct](http://msdn.microsoft.com/es-es/library/ah19swz4.aspx) pueda implementar.

Mediante las interfaces, puede incluir, por ejemplo, comportamiento de varios orígenes en una clase. Esa función es importante en C# porque el lenguaje no admite la herencia múltiple de clases. Además, debe usar una interfaz para simular la herencia para los structs, porque no pueden heredar realmente de otro struct o clase.

Define una interfaz mediante la palabra clave de [interfaz](http://msdn.microsoft.com/es-es/library/87d83y5b.aspx), como se muestra en el ejemplo siguiente.

C#

interface IEquatable<T>

{

bool Equals(T obj);

}

Cualquier clase o struct que implemente la interfaz de [IEquatable<T>](http://msdn.microsoft.com/es-es/library/ms131187.aspx) debe contener una definición de un método de [Equals](http://msdn.microsoft.com/es-es/library/ms131190.aspx) que coincida con la firma que la interfaz especifica. Como resultado, puede contar en una clase que implemente IEquatable<T>para contener un método de Equals con el que una instancia de la clase puede determinar si es igual a otra instancia de la misma clase.

La definición de IEquatable<T> no proporciona una implementación para Equals. Esta interfaz define sólo la firma. De esa manera, una interfaz en C# es similar a una clase abstracta en la que todos los métodos son abstractos. Sin embargo, una clase o struct puede implementar varias interfaces, pero una clase puede heredar una única clase, implícita o no. Por consiguiente, mediante las interfaces, puede incluir comportamiento de varios orígenes en una clase.

Para obtener más información sobre las clases abstractas, vea [Clases y miembros de clase abstractos y sealed](http://msdn.microsoft.com/es-es/library/ms173150.aspx).

Las interfaces pueden contener métodos, propiedades, eventos, indizadores, o cualquier combinación de estos tipos de cuatro miembros. Para obtener vínculos a los ejemplos, vea [Secciones relacionadas](http://msdn.microsoft.com/es-es/library/ms173156.aspx#bkmk_relatedsections). Una interfaz no puede contener constantes, campos, operadores, constructores de instancia, destructores, o tipos. Los miembros de interfaz son automáticamente públicos, y no pueden incluir ninguna modificadores de acceso. Los miembros tampoco pueden ser [estático](http://msdn.microsoft.com/es-es/library/98f28cdx.aspx).

Para implementar un miembro de interfaz, el miembro correspondiente de la clase que implementa debe ser público, no estático, y tiene el mismo nombre y firma que el miembro de interfaz.

Cuando una clase o struct implementa una interfaz, clase o struct debe proporcionar una implementación para todos los miembros que la interfaz define. La propia interfaz no proporciona ninguna funcionalidad que una clase o struct puede heredar de la manera que puede heredar funcionalidad de la clase base. Sin embargo, si una clase base implementa una interfaz, cualquier clase que se deriva de la clase base hereda esa implementación.

El ejemplo siguiente se muestra una implementación de la interfaz de IEquatable<T>.La clase que implementa, Car, debe proporcionar una implementación del método de[Equals](http://msdn.microsoft.com/es-es/library/ms131190.aspx).

C#

public class Car : IEquatable<Car>

{

public string Make {get; set;}

public string Model { get; set; }

public string Year { get; set; }

// Implementation of IEquatable<T> interface

public bool Equals(Car car)

{

if (this.Make == car.Make &&

this.Model == car.Model &&

this.Year == car.Year)

{

return true;

}

else

return false;

}

}

Las propiedades y los indizadores de una clase pueden definir los descriptores de acceso adicionales para una propiedad o indizador que se define en una interfaz. Por ejemplo, una interfaz puede declarar una propiedad que tiene un descriptor de[obtenga](http://msdn.microsoft.com/es-es/library/ms228503.aspx). La clase que implementa la interfaz puede declarar la misma propiedad con**get** y el descriptor de [establecer](http://msdn.microsoft.com/es-es/library/ms228368.aspx). Sin embargo, si la propiedad o el indizador utiliza una implementación explícita, los descriptores de acceso deben coincidir. Para obtener más información sobre la implementación explícita, vea [Implementación de interfaz explícita (Guía de programación de C#)](http://msdn.microsoft.com/es-es/library/ms173157.aspx) y [Propiedades de interfaces (Guía de programación de C#)](http://msdn.microsoft.com/es-es/library/64syzecx.aspx).

Las interfaces pueden implementar otras interfaces. Una clase podría incluir varias veces una interfaz a través de las clases base que hereda o las interfaces que otras interfaces implementadas. Sin embargo, la clase puede proporcionar una implementación de una interfaz sólo una vez y solo si la clase declara la interfaz como parte de la definición de clase (class ClassName : InterfaceName). Si se hereda la interfaz porque se heredó una clase base que implementa la interfaz, la clase base proporciona la implementación de los miembros de interfaz. Sin embargo, la clase derivada puede volver a implementar los miembros de interfaz en lugar de utilizar la implementación heredada.

Una clase base puede implementar también los miembros de la interfaz con los miembros virtuales. En ese caso, una clase derivada puede cambiar el comportamiento de la interfaz al invalidar los miembros virtuales. Para obtener más información sobre miembros virtuales, vea [Polimorfismo (Guía de programación de C#)](http://msdn.microsoft.com/es-es/library/ms173152.aspx).

[Interfaces de resumen](javascript:void(0))

Una interfaz tiene las siguientes propiedades:

* Una interfaz es como una clase base abstracta. Cualquier clase o struct que implemente la interfaz debe implementar todos sus miembros.
* Una interfaz no puede ser instanciada directamente. Cualquier clase o struct implementan sus miembros que implementa la interfaz.
* Las interfaces pueden contener eventos, métodos, indizadores y propiedades.
* Las interfaces no contienen implementaciones de métodos.
* Una clase o struct puede implementar varias interfaces. Una clase puede heredar una clase base y también implementar una o más interfaces.