Modificadores de acceso (Guía de programación de C#)

Artículo • 05/12/2023

Todos los tipos y miembros de tipo tienen un nivel de accesibilidad. El nivel de accesibilidad controla si se pueden usar desde otro código del ensamblado u otros ensamblados. Un ensamblado es un archivo *.dll* o *.exe* creado mediante la compilación de uno o varios archivos *.cs* en una sola compilación. Use los modificadores de acceso siguientes para especificar la accesibilidad de un tipo o miembro cuando lo declare:

- public: Puede obtener acceso al tipo o miembro cualquier otro código del mismo ensamblado o de otro ensamblado que haga referencia a éste. El nivel de accesibilidad de los miembros públicos de un tipo se controla mediante el nivel de accesibilidad del propio tipo.
- private: solamente el código de la misma `class` o `struct` puede acceder al tipo o miembro.
- protected: solamente el código de la misma `class`, o bien de una `class` derivada de esa `class`, puede acceder al tipo o miembro.
- internal: Puede obtener acceso al tipo o miembro cualquier código del mismo ensamblado, pero no de un ensamblado distinto. En otras palabras, se puede acceder a tipos o miembros `internal` desde el código que forma parte de la misma compilación.
- protected internal: cualquier código del ensamblado en el que se ha declarado, o desde una `class` derivada de otro ensamblado, puede acceder al tipo o miembro.
- private protected: se puede tener acceso al tipo o miembro mediante tipos derivados del objeto `class` que se declaran dentro de su ensamblado contenedor.

Tabla de resumen

Expandir tabla

Ubicación del autor de la llamada	`public`	`protected internal`	`protected`	`internal`	`private protected`	`private`
Dentro de la clase	✓	✓	✓	✓	✓	✓

Ubicación del autor de la llamada	`public`	`protected internal`	`protected`	`internal`	`private protected`	`private`
Clase derivada (mismo ensamblado)	✓	✓	✓	✓	✓	✗
Clase no derivada (mismo ensamblado)	✓	✓	✗	✓	✗	✗
Clase derivada (otro ensamblado)	✓	✓	✓	✗	✗	✗
Clase no derivada (otro ensamblado)	✓	✗	✗	✗	✗	✗

En los ejemplos siguientes se muestra cómo especificar modificadores de acceso en un tipo y miembro:

C#

```
public class Bicycle
{
    public void Pedal() { }
}
```

No todos los modificadores de acceso son válidos para todos los tipos o miembros de todos los contextos. En algunos casos, la accesibilidad de un miembro de tipo está restringida por la accesibilidad de su tipo contenedor.

Accesibilidad de clases, registros y estructuras

Las clases, los registros y las estructuras que se declaran directamente en un espacio de nombres (es decir, que no están anidadas en otras clases o estructuras) pueden ser `public` o `internal`. Si no se especifica ningún modificador de acceso, el valor predeterminado es `internal`.

Los miembros de estructura, incluidas las clases y las estructuras anidadas, se pueden declarar como `public`, `internal` o `private`. Los miembros de clase, incluidas las clases y las estructuras anidadas, pueden ser `public`, `protected internal`, `protected`, `internal`, `private protected` o `private`. Los miembros de clase y estructura, incluidas las clases y las

estructuras anidadas, tienen acceso `private` de forma predeterminada. Los tipos anidados privados no son accesibles desde fuera del tipo contenedor.

Las clases derivadas y los registros derivados no pueden tener mayor accesibilidad que sus tipos base. No se puede declarar una clase pública `B` que derive de una clase interna `A`. Si se permitiera, convertiría `A` en público, porque todos los miembros `protected` o `internal` de `A` son accesibles desde la clase derivada.

Puede habilitar otros ensamblados concretos para acceder a los tipos internos mediante `InternalsVisibleToAttribute`. Para más información, vea Ensamblados de confianza.

Accesibilidad de miembros de clases, registros y estructuras

Los miembros de clases y registros (incluidas las clases, los registros y las estructuras anidados) se pueden declarar con cualquiera de los seis tipos de acceso. Los miembros de estructura no se pueden declarar como `protected`, `protected internal` o `private protected` porque las estructuras no admiten la herencia.

Normalmente, la accesibilidad de un miembro no es mayor que la del tipo que lo contiene. Pero un miembro `public` de una clase interna podría ser accesible desde fuera del ensamblado si el miembro implementa los métodos de interfaz o invalida los métodos virtuales definidos en una clase base pública.

El tipo de cualquier miembro que sea un campo, propiedad o evento debe ser al menos tan accesible como el propio miembro. Del mismo modo, el tipo devuelto y los tipos de parámetro de cualquier método, indizador o delegado deben ser al menos tan accesibles como el propio miembro. Por ejemplo, no puede tener un método `public M` que devuelva una clase `C` a menos que `C` también sea `public`. Del mismo modo, no puede tener una propiedad `protected` de tipo `A` si `A` se declara como `private`.

Los operadores definidos por el usuario siempre se deben declarar como `public` y `static`. Para obtener más información, vea Sobrecarga de operadores.

Los finalizadores no pueden tener modificadores de accesibilidad.

Para establecer el nivel de acceso de un miembro de `class`, `record` o `struct`, agregue la palabra clave adecuada a la declaración de miembro, como se muestra en el ejemplo siguiente.

C#

```
// public class:
public class Tricycle
{
    // protected method:
    protected void Pedal() { }

    // private field:
    private int _wheels = 3;

    // protected internal property:
    protected internal int Wheels
    {
        get { return _wheels; }
    }
}
```

Otros tipos

Las interfaces declaradas directamente en un espacio de nombres pueden ser `public` o `internal` y, al igual que las clases y las estructuras, su valor predeterminado es el acceso `internal`. Los miembros de interfaz son `public` de manera predeterminada porque el propósito de una interfaz es permitir que otros tipos accedan a una clase o estructura. Las declaraciones de miembros de interfaz pueden incluir cualquier modificador de acceso. Esto es muy útil para que los métodos estáticos proporcionen implementaciones comunes necesarias para todos los implementadores de una clase.

Los miembros de enumeración siempre son `public` y no se les puede aplicar ningún modificador de acceso.

Los delegados se comportan como las clases y las estructuras. De forma predeterminada, tienen acceso `internal` cuando se declaran directamente en un espacio de nombres y acceso `private` cuando están anidados.

Tabla de resumen de acceso predeterminada

Expandir tabla

Tipo	Acceso predeterminado
`class`	internal
`struct`	internal
`interface`	internal
`record`	internal
`enum`	internal
`interface` miembros	pública
Tipos anónimos (Guía de programación de C#).	internal
Miembros de clase, registro y estructura	private

Para más información, consulte la página Niveles de accesibilidad.

Especificación del lenguaje C#

Para obtener más información, consulte la Especificación del lenguaje C#. La especificación del lenguaje es la fuente definitiva de la sintaxis y el uso de C#.

Vea también

- Especificación del orden del modificador (regla de estilo IDE0036)
- Guía de programación de C#
- El sistema de tipos de C#
- Interfaces
- Niveles de accesibilidad
- private
- public
- internal
- protected
- protected internal
- private protected
- sealed
- clase
- struct

- interface
- Tipos anónimos (Guía de programación de C#).

Colaborar con nosotros en GitHub

El origen de este contenido se puede encontrar en GitHub, donde también puede crear y revisar problemas y solicitudes de incorporación de cambios. Para más información, consulte nuestra guía para colaboradores.

Comentarios de .NET

.NET es un proyecto de código abierto. Seleccione un vínculo para proporcionar comentarios:

Abrir incidencia con la documentación

Proporcionar comentarios sobre el producto