

Materia: Programación visual III.

Profesor: Ing. Daniel Nuño.

Tarea 14: Eventos y Delegados.

Alumno: René Alejandro Barrera Ramírez.

INTRODUCCIÓN

En el presente documento se realiza un resumen de un sitio web donde contiene información acerca de los eventos y delegados, en términos de programación. Entre toda la información se explica la diferencia que existe entre ellos y como se utilizan.

Delegados

Algunas veces se usa para referirse a un tipo de delegado , y en otras ocasiones puede usarse para referirse a una instancia de un tipo de delegado.

Tipos de delegado:

Los tipos de delegados se declaran con la palabra clave **delegate**

De alguna manera, puede pensar en un tipo de delegado como una interfaz con un solo método. Especifica la firma de un método, y cuando tiene una instancia de delegado, puede llamarlo como si fuera un método con la misma firma. Los delegados proporcionan otras características, pero la capacidad de realizar llamadas con una firma particular es la razón de la existencia del concepto de delegado. Los delegados tienen una referencia a un método y una referencia al objeto de destino al que se debe llamar el método.

Tenga en cuenta que la palabra clave delegateno siempre significa que se está declarando un tipo de delegado. Se utiliza la misma palabra clave al crear instancias del tipo delegado utilizando métodos anónimos.

**Crear instancias delegadas**

Usamos lo que la especificación de C # llama una expresión de creación de delegado que tiene la forma *new delegate-type (expression)*. La expresión debe ser otro delegado del mismo tipo o un grupo de métodos : el nombre de un método y, opcionalmente, un destino, especificado como si estuviera llamando al método, pero sin los argumentos o soportes.

Un punto a tener en cuenta es que las instancias delegadas pueden referirse a métodos y objetivos que normalmente no serían visibles en el punto en que se realiza la llamada. Por ejemplo, se puede usar un método privado para crear una instancia de delegado, y luego la instancia de delegado se puede devolver de un miembro público. Alternativamente, el objetivo de una instancia puede ser un objeto del que la persona que llama eventualmente no sabe nada. Sin embargo, tanto el objetivo como el método deben ser accesibles para el código de creación . En otras palabras, si (y solo si) puede llamar a un método en particular sobre un objeto en particular, puede usar ese método y destino para la creación de delegados. Los derechos de acceso se ignoran efectivamente en el momento de la llamada.

**Llamar instancias delegadas**

Las instancias de delegado se llaman como si fueran los propios métodos. Por ejemplo:

string result = d1(10);

El método al que hace referencia la instancia de delegado se llama en el objeto de destino (si hay uno) y se devuelve el resultado. Producir un programa completo para demostrar esto sin incluir una gran cantidad de código aparentemente irrelevante es complicado.

**Delegados combinados**

Los delegados se pueden combinar de modo que cuando llame al delegado, se llame a una lista completa de métodos, potencialmente con diferentes objetivos.

Es importante comprender que las instancias delegadas siempre son inmutables. Cualquier cosa que los combine (o quite uno del otro) crea una nueva instancia de delegado para representar la nueva lista de objetivos / métodos para llamar.

La combinación de dos instancias de delegado generalmente se realiza utilizando el operador de suma, como si las instancias de delegado fueran cadenas o números. Restar uno de otro generalmente se hace con el operador de resta. Tenga en cuenta que cuando resta un delegado combinado de otro, la resta funciona en términos de listas . Si la lista para restar no se encuentra en la lista original, el resultado es solo la lista original. De lo contrario, se elimina la última aparición de la lista.

**Eventos**

Creo que la forma más fácil de entender los eventos es pensar en ellos un poco como propiedades. Si bien las propiedades parecen ser campos, definitivamente no lo son, y puede escribir propiedades que no usan campos en absoluto. Del mismo modo, si bien los eventos parecen instancias delegadas en términos de la forma en que expresa las operaciones de agregar y quitar, no lo son.

Los eventos son pares de métodos, adecuadamente decorados en IL para unirlos y hacer que los idiomas sepan que los métodos representan eventos. Los métodos corresponden a operaciones de agregar y quitar , cada una de las cuales toma un parámetro de instancia de delegado del mismo tipo (el tipo del evento). Lo que haga con esas operaciones depende de usted, pero el uso típico es agregar o eliminar el delegado de una lista de controladores para el evento. Cuando se desencadena el evento (cualquiera que sea ese desencadenante: un clic en el botón, un tiempo de espera, una excepción no controlada), se llama a los controladores a su vez.

Los eventos en sí pueden ser declarados de dos maneras. El primero es con métodos explícitos de agregar y quitar, declarados de manera muy similar a las propiedades, pero con la palabra clave **event**.

## **Un atajo: eventos de campo**

C # proporciona una manera simple de declarar una variable delegada y un evento al mismo tiempo. Esto se llama un *evento similar a* un *campo* y se declara de manera muy simple: es lo mismo que la declaración de evento "a largo plazo", pero sin la parte "cuerpo":

public event EventHandler MyEvent;

Esto crea una variable delegada y un evento, ambos con el mismo tipo. El acceso al evento está determinado por la declaración del evento (por lo que el ejemplo anterior crea un evento público, por ejemplo) pero la variable delegada siempre es privada.

**Eventos seguros para subprocesos**

Si desea ser verdaderamente seguro para subprocesos, de modo que cuando genera un evento siempre usa el valor más reciente de la variable delegada, además de asegurarse de que las operaciones de agregar / quitar no interfieran entre sí, debe escriba el cuerpo de las operaciones de agregar / quitar usted mismo.

Puede usar un solo candado para todos sus eventos, o incluso para otras cosas, depende de su situación. Tenga en cuenta que debe asignar el valor actual a una variable local dentro del bloqueo (para obtener el valor más reciente) y luego probarlo para determinar la nulidad y ejecutarlo fuera del bloqueo: mantener el bloqueo mientras se genera el evento es una muy mala idea, como fácilmente podrías llegar a un punto muerto. (Es posible que los controladores de eventos necesiten esperar a que otro hilo haga algo, y si ese otro hilo llamara a la operación de agregar o quitar en su evento, obtendría un punto muerto).

**Delegar instancias: otros métodos**

Estos son opcionales en lo que respecta a la especificación CLI, pero los tipos de delegado de C # siempre los proporcionan. Siguen el mismo modelo para la ejecución asincrónica que el resto de .NET, lo que permite proporcionar un controlador de devolución de llamada, junto con un objeto para almacenar información de estado. Los delegados se ejecutan en subprocesos creados por el grupo de subprocesos del sistema.

**CONCLUSIÓN**

Los delegados nos ayudan a reutilizar métodos de otras clases sin la necesidad de hacer herencia, esto es muy útil en la práctica ya que no le asignamos *n* cantidad de métodos provenientes de la clase desde donde se hará la referencia al método en específico.