****

**Hans Castillo Solis**

**Ingenieria de software**

**4.-A T/M**

**Daniel nuno**

**Tarea PVT14**

**RESUMEN DELEGADOS Y EVENTOS**

## Tipos de delegado

Los tipos de delegados se declaran con la delegate palabra clave. Pueden aparecer solos o anidados dentro de una clase.

namespace DelegateArticle

{

public delegate string FirstDelegate (int x);

public class Sample

{

public delegate void SecondDelegate (char a, char b);

}

}

Este código declara dos tipos de delegado. El primero es el DelegateArticle.FirstDelegate que tiene un único parámetro de tipo int y devuelve a string. El segundo es el DelegateArticle.Sample.SecondDelegate que tiene dos char parámetros y no devuelve nada (porque el tipo de retorno se especifica como void).

Cualquier tipo de delegado se crea tiene los miembros heredados de sus tipos de padres, un constructor con parámetros de object e IntPtry tres métodos adicionales: Invoke, BeginInvokey EndInvoke.

**Delegar instancias**

Para los métodos estáticos, no se requiere un objetivo. El CLR en sí mismo admite otras formas de delegado ligeramente diferentes, donde el primer argumento pasado a un método estático se mantiene dentro del delegado, o el objetivo de un método de instancia se proporciona como un argumento cuando se llama al método. Consulte la documentación para System.Delegate obtener más información sobre esto si está interesado, pero no se preocupe demasiado por eso.

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.

**Llamar instancias delegadas**

Las instancias de delegado se llaman como si fueran los propios métodos. 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. Sin embargo, aquí hay un programa que da un ejemplo de un método estático y uno de un método de instancia. DelegateTest.StaticMethod se podría escribir como se acaba StaticMethodde la misma manera que (dentro de un método de instancia) se podría escribir InstanceMethoden lugar de this.InstanceMethod- He incluido el nombre de la clase sólo para que quede claro cómo se haría referencia a los métodos de otras clases.

**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. Cuando dije antes que un delegado contenía un objetivo y un método, eso fue una ligera simplificación. Eso es lo que contiene una instancia de delegado que representa un método. En aras de la claridad, me referiré a tales instancias de delegado como *delegados simples*. La alternativa es una instancia de delegado que es efectivamente una lista de delegados simples, todos del mismo tipo (es decir, que tienen la misma firma). Llamaré a estos *delegados combinados*. Los delegados combinados pueden combinarse entre sí, creando efectivamente una gran lista de delegados simples de la manera obvia. 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.

**Eventos**

Los eventos no son instancias delegadas. 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. Tenga en cuenta que en C #, la llamada de los controladores de eventos *no forma* parte del evento en sí. (CIL define una asociación con un raise\_*eventName* método y, de hecho, "otros" métodos, pero estos no se usan en C #). 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 event palabra clave. Aquí hay un ejemplo de un evento para el System.EventHandlertipo de delegado. Tenga en cuenta que en realidad no hace nada con las instancias de delegado que se pasan a los métodos de agregar y quitar, solo imprime qué operación se ha llamado. Tenga en cuenta que la operación de eliminación se llama aunque le hayamos dicho que se elimine null.

**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.