Métodos asincrónicos

Mediante la característica asincrónica, puede invocar métodos asincrónicos sin usar definiciones de llamada explícitas ni dividir manualmente el código en varios métodos o expresiones lambda.

Si marca un método con el modificador async , puede usar el operador await en el método. Cuando el control alcanza una expresión await en el método asincrónico, el control se devuelve al autor de llamada y se progreso del método se suspende hasta que se completa la tarea esperada. Cuando se completa la tarea, la ejecución puede reanudarse en el método.

Un método asincrónico vuelve al autor de llamada cuando encuentra el primer objeto esperado que aún no se ha completado o cuando llega al final del método asincrónico, lo que ocurra primero.

Un método asincrónico puede tener un tipo de valor devuelto de Task<TResult>, Task, o nulo. El tipo de valor devuelto nulo se utiliza principalmente para definir controladores de eventos, donde se requiere un tipo de valor devuelto nulo. No se puede esperar un método asincrónico que devuelve nulo, y el autor de llamada de un método con tipo de devolución nulo no puede capturar ninguna excepción producida por este.

En el ejemplo siguiente, DelayAsync es un método asincrónico con un tipo de valor devuelto de Task<TResult>. DelayAsync tiene una instrucción return que devuelve un entero. Por lo tanto, la declaración del método de DelayAsync debe tener un tipo de valor devuelto de Task<int>. Dado que el tipo de valor devuelto es Task<int>, la evaluación de la expresión await en DoSomethingAsync genera un entero, como se demuestra en la siguiente instrucción: int result = await delayTask.

El método `startButton_Click` es un ejemplo de un método asincrónico con un tipo de valor devuelto nulo. Dado que `DoSomethingAsync` es un método asincrónico, la tarea de la llamada a `DoSomethingAsync` debe esperar, como se muestra en la siguiente instrucción: `await DoSomethingAsync();`. El método `startButton_Click` debe definirse con el modificador `async` porque el método tiene una expresión `await` .

```
// using System.Diagnostics;
// using System.Threading.Tasks;

// This Click event is marked with the async modifier.
private async void startButton_Click(object sender, RoutedEventArgs e)
{
```

```
    await DoSomethingAsync();
}

private async Task DoSomethingAsync()
{
    Task<int> delayTask = DelayAsync();
    int result = await delayTask;

    // The previous two statements may be combined into
    // the following statement.
    //int result = await DelayAsync();

    Debug.WriteLine("Result: " + result);
}

private async Task<int> DelayAsync()
{
    await Task.Delay(100);
    return 5;
}

// Output:
//  Result: 5
```

Un método asincrónico no puede declarar ningún parámetro ref u out , pero puede llamar a los métodos que tienen estos parámetros.