Introducción a las Tuplas

La clase Tuple <T> se introdujo en .NET Framework 4.0. Una tupla es una estructura de datos que contiene una secuencia de elementos de diferentes tipos de datos. Puede usarse donde neceselementoos tener una estructura de datos para contener un objeto con propiedades, pero donde no queremos crear un tipo separado para él.

```
Tuple<T1, T2, T3, T4, T5, T6, T7, TRest>
```

El siguiente ejemplo crea una tupla con tres elementos:

Ejemplo: Tuple

```
Tuple<int, string, string> persona = new Tuple <int, string,
string>(1, "Juan", "Gómez");
```

En el ejemplo anterior, creamos una instancia de Tuple que contiene un registro de una persona. Especificamos un tipo para cada elemento y pasamos valores al constructor. Especificar el tipo de cada elemento es engorroso. Por lo tanto, C # incluye una clase auxiliar Tuple estática que devuelve una instancia de Tuple <T> sin especificar el tipo de cada elemento, como se muestra a continuación.

```
var persona = Tuple.Create(1, "Juan", "Gómez");
```

Una tupla solo puede incluir un máximo de ocho elementos. Da un error de compilación cuando intentas incluir más de ocho elementos.

```
var numeros = Tuple.Create(1, 2, 3, 4, 5, 6, 7, 8);
```

Accediendo a los elementos de una Tupla

Se puede acceder a los elementos de una tupla con las propiedades Elemento <numeroElemento>, por ejemplo, Elemento1, Elemento2, Elemento3 y así sucesivamente hasta la propiedad Elemento7. La propiedad Elemento1 devuelve el primer elemento, Elemento2 devuelve el segundo elemento y así sucesivamente. El último elemento (el 8º elemento) se devolverá utilizando la propiedad Rest.

Ejemplo: acceder a los elementos de la Tupla

```
var persona = Tuple.Create(1, "Juan", "Gómez");
persona.Elemento1; // devuelve 1
persona.Elemento2; // devuelve "Juan"
persona.Elemento3; // devuelve "Gómez"
```

```
var numeros = Tuple.Create("Uno", 2, 3, "Cuatro", 5, "Seis", 7, 8);
numeros.Elemento1; // returns "Uno"
numeros.Elemento2; // devuelve 2
numeros.Elemento3; // devuelve 3
numeros.Elemento4; // devuelve "Cuatro"
numeros.Elemento5; // devuelve 5
numeros.Elemento6; // devuelve "Seis"
numeros.Elemento7; // devuelve 7
numeros.Rest; // devuelve (8)
numeros.Rest.Elemento1; // devuelve 8
```

En general, la 8ª posición es para la tupla anidada a la que puede acceder utilizando la propiedad Rest.

Tuplas anidadas:

Si deseamos incluir más de ocho elementos en una tupla, podemos hacerlo anidando otro objeto de tupla como el octavo elemento. Se puede acceder a la última tupla anidada usando la propiedad Rest. Para acceder al elemento de la tupla anidada, usamos la propiedad Rest.Elemento1.Elemento <elnumeroElemento>.

Ejemplo: tupla anidada

```
var numeros = Tuple.Create(1, 2, 3, 4, 5, 6, 7, Tuple.Create(8, 9, 10, 11,
12, 13));
numeros.Elemento1; // devuelve 1
numeros.Elemento7; // devuelve 7
numeros.Rest.Elemento1; // devuelve (8, 9, 10, 11, 12, 13)
numeros.Rest.Elemento1.Elemento1; // devuelve 8
numeros.Rest.Elemento1.Elemento2; // devuelve 9
```

Puede incluir el objeto de tupla anidado en cualquier lugar de la secuencia. Sin embargo, se recomienda colocar la tupla anidada al final de la secuencia para que se pueda acceder utilizando la propiedad Rest.

```
var numeros = Tuple.Create(1, 2, Tuple.Create(3, 4, 5, 6, 7,  8), 9, 10, 11,
12, 13 );
numeros.Elemento1; // devuelve 1
numeros.Elemento2; // devuelve 2
numeros.Elemento3; // devuelve (3, 4, 5, 6, 7, 8)
numeros.Elemento3.Elemento1; // devuelve 3
numeros.Elemento4; // devuelve 9
numeros.Rest.Elemento1; // devuelve 13
```

Tuple as a Method Parameter:

Los métodos pueden tener una tupla como un parámetro.

```
static void Main(string[] args)
```

```
{
    var persona = Tuple.Create(1, "Juan", "Gómez");
    MostrarTupla(persona);
}

static void MostrarTupla(Tuple<int,string,string> persona)
{
    Console.WriteLine($"Id = {persona.Elemento1}");
    Console.WriteLine($"Nombre = {persona.Elemento2}");
    Console.WriteLine($"Apellidos = {persona.Elemento3}");
}
```

Tupla como Tipo de Retorno/Devolución:

Una Tupla puede ser usada desde un método.

```
static void Main(string[] args)
{
    var persona = GetPersona();
}

static Tuple<int, string, string> GetPersona()
{
    return Tuple.Create(1, "Manuel", "García");
}
```

Principales Usos de las Tuplas:

- Cuando necesitemos devolver múltiples valores de un método sin usar los parámetros ref o out.
- Cuando queremos pasar múltiples valores a un método a través de un solo parámetro.
- Cuando necesitemos mantener un registro de la base de datos o algunos valores temporalmente sin crear una clase separada.

Limitaciones de las Tuplas:

- Tupla es un tipo de referencia y no un tipo de valor. Se asigna en la memoria heap y podría dar lugar a operaciones intensivas de la CPU.
- Tupla está limitada a incluir 8 elementos. Debemos usar tuplas anidadas si necesitamos almacenar más elementos. Sin embargo, esto puede generar ambigüedad.
- Se puede acceder a los elementos Tuple usando propiedades con un patrón de nombre_Elemento <numeroElemento> lo cual no tiene mucho sentido.

FIN