عنوان: **Tuple در دات نت 4**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۲/۱۷ ۱۳:۰۸:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: #C

نوع جدیدی در دات نت 4 به نام Tuple اضافه شده است که در این مطلب به بررسی آن خواهیم پرداخت.
در ریاضیات، Tuple به معنای لیست مرتبی از اعضاء با تعداد مشخص است. Tuple در زبان‌های برنامه نویسی Dynamic مانند اف شارپ، LISP ، Perl و بسیاری موارد دیگر مطلب جدیدی نیست. در زبان‌های dynamic برنامه نویس‌ها می‌توانند متغیرها را بدون معرفی نوع آن‌ها تعریف کنند. اما در زبان‌های Static مانند سی شارپ ، برنامه نویس‌ها موظفند نوع متغیرها را پیش از کامپایل آن‌ها معرفی کنند که هر چند کار کد نویسی را اندکی بیشتر می‌کند اما به این صورت شاهد خطاهای کمتری نیز خواهیم بود (البته سی شارپ 4 این مورد را با معرفی واژه‌ی کلیدی dynamic تغییر داده است).
برای مثال در اف شارپ داریم:

```
let data = (“John Doe”, 42)
```

که سبب ایجاد یک tuple که المان اول آن یک رشته و المان دوم آن یک عدد صحیح است می‌شود. اگر data را بخواهیم نمایش دهیم خروجی آن به صورت زیر خواهد بود:

```
printf “%A” data
// Output: (“John Doe”,42)
```

در دات نت 4 فضای نام جدیدی به نام System.Tuple معرفی شده است که در حقیقت ارائه دهنده‌ی نوعی جنریک می‌باشد که توانایی در برگیری انواع مختلفی را دارا است :

```
public class Tuple<T1>
up to:
public class Tuple<T1, T2, T3, T4, T5, T6, T7, TRest>
```

همانند آرایه‌ها، اندازه‌ی Tuples نیز پس از تعریف قابل تغییر نیستند (immutable). اما تفاوت مهم آن با یک آرایه در این است که اعضای آن می‌توانند نوع‌های کاملا متفاوتی داشته باشند. همچنین تفاوت مهم آن با یک ArrayList یا آرایه‌ای از نوع Object، مشخص بودن نوع هر یک از اعضاء آن است که type safety بیشتری را به همراه خواهد داشت و کامپایلر می‌تواند در حین کامپایل دقیقا مشخص نماید که اطلاعات دریافتی از نوع صحیحی هستند یا خیر.

یک مثال کامل از Tuples را در کلاس زیر ملاحظه خواهید نمود:

```
using System;
using System.Linq;
using System.Collections.Generic;

namespace TupleTest
{
   class TupleCS4
   {
  #region Methods (4)

  // Public Methods (4)

        public static Tuple<string, string> GetFNameLName(string name)
        {
            if (string.IsNullOrWhiteSpace(name))
```

```
                throw new NullReferenceException("name is empty.");

            var nameParts = name.Split(',');

            if (nameParts.Length != 2)
                throw new FormatException("name must contain ','");

            return Tuple.Create(nameParts[0], nameParts[1]);
        }

        public static void PrintSelectedTuple()
        {
            var list = new List<Tuple<string, int>>
                        {
                            new Tuple<string, int>("A", 1),
                            new Tuple<string, int>("B", 2),
                            new Tuple<string, int>("C", 3)
                        };

            var item = list.Where(x => x.Item2 == 2).SingleOrDefault();
            if (item != null)
                Console.WriteLine("Selected Item1: {0}, Item2: {1}",
                    item.Item1, item.Item2);
        }

        public static void PrintTuples()
        {
            var tuple1 = new Tuple<int>(12);
            Console.WriteLine("tuple1 contains: item1:{0}", tuple1.Item1);

            var tuple2 = Tuple.Create("Item1", 12);
            Console.WriteLine("tuple2 contains: item1:{0}, item2:{1}",
                tuple2.Item1, tuple2.Item2);

            var tuple3 = Tuple.Create(new DateTime(2010, 5, 6), "Item2", 20);
            Console.WriteLine("tuple3 contains: item1:{0}, item2:{1}, item3:{2}",
                tuple3.Item1, tuple3.Item2, tuple3.Item3);
        }

        public static void Tuple8()
        {
            var tup =
                new Tuple<int, int, int, int, int, int, int, Tuple<int, int>>
                    (1, 2, 3, 4, 5, 6, 7, new Tuple<int, int>(8, 9));

            Console.WriteLine("tup.Rest Item1: {0}, Item2: {1}",
                    tup.Rest.Item1,tup.Rest.Item2);
        }

  #endregion Methods
    }
}
```

```
using System;

namespace TupleTest
{
   class Program
   {
        static void Main()
        {
            var data = TupleCS4.GetFNameLName("Vahid, Nasiri");
            Console.WriteLine("Data Item1:{0} & Item2:{1}",
                    data.Item1, data.Item2);

            TupleCS4.PrintTuples();

            TupleCS4.PrintSelectedTuple();

            TupleCS4.Tuple8();

            Console.WriteLine("Press a key...");
            Console.ReadKey();
        }
   }
}
```

توضیحات :

- روش‌های متفاوت ایجاد Tuples را در متد PrintTuples می‌توانید ملاحظه نمائید. همچنین نحوه‌ی دسترسی به مقادیر هر کدام از اعضاء نیز مشخص شده است.

- کاربرد مهم Tuples در متد GetFNameLName نمایش داده شده است؛ زمانیکه نیاز است تا چندین خروجی از یک تابع داشته باشیم. به این صورت دیگر نیازی به تعریف آرگومان‌هایی به همراه واژه کلیدی out نخواهد بود یا دیگر نیازی نیست تا یک شیء جدید را ایجاد کرده و خروجی را به آن نسبت دهیم. به همان سادگی زبان‌های dynamic در اینجا نیز می‌توان یک tuple را ایجاد و استفاده کرد.

- بدیهی است از Tuples در یک لیست جنریک و یا حالات دیگر نیز می‌توان استفاده کرد. مثالی از این دست را در متد PrintSelectedTuple ملاحظه خواهید نمود. ابتدا یک لیست جنریک از Tuple ایی با دو عضو تشکیل شده است. سپس با استفاده از امکانات LINQ ، عضوی که آیتم دوم آن مساوی 2 است یافت شده و سپس المان‌های آن نمایش داده می‌شود.

- نکته‌ی دیگری را که حین کار با Tuples می‌توان در نظر داشت این است که اعضای آن حداکثر شامل 8 عضو می‌توانند باشند که عضو آخر باید یک Tuple تعریف گردد و بدیهی است این Tuple نیز می‌تواند شامل 8 عضو دیگر باشد و الی آخر که نمونه‌ای از آن را در متد Tuple8 می‌توان مشاهده کرد.

نظرات خوانندگان

نویسنده: Afshar Mohebbi
تاریخ: ۱۳۸۹/۰۲/۱۷ ۱۶:۱۸:۳۲

جالب و مفید بود. ممنون.

نویسنده: علی اقدم
تاریخ: ۱۳۸۹/۱۱/۰۳ ۲۲:۵۷:۱۰

البته در استفاده از Tuple ها باید به این نکته توجه نمود که بعد از نمونه سازی یک Tuple دیگر امکان تغییر خصوصیات اون نمونه وجود نداره که بدلیل عدم وجود setter است،
همچنین می توان از خصوصیت Rest در Tuple های تودرتو استفاده نمود، مثلا

```
tup.Rest.Rest.Item1
```