Bài 11: TUPLE TRONG C#

Xem bài học trên website để ủng hộ Kteam: Tuple trong C#

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage How Kteam nhé!

Dẫn nhập

Ở các bài học trước, chúng ta đã cùng nhau tìm hiểu về DICTIONARY TRONG C#. Hôm nay chúng ta sẽ cùng tìm hiểu về **Tuple trong C#**.

Nội dung

Để đọc hiểu bài này tốt nhất các bạn nên có kiến thức cơ bản về các phần:

- BIẾN , KIỂU DỮ LIỆU, TOÁN TỬ trong C#
- CÂU ĐIỀU KIỆN trong C#
- Cấu trúc cơ bản của VÒNG LẶP, HÀM trong C#
- MẢNG trong C#
- LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG C#

Trong bài học này, chúng ta sẽ cùng tìm hiểu các vấn đề:

- Tuple là gì?
- Cách sử dụng Tuple.

Tuple là gì?

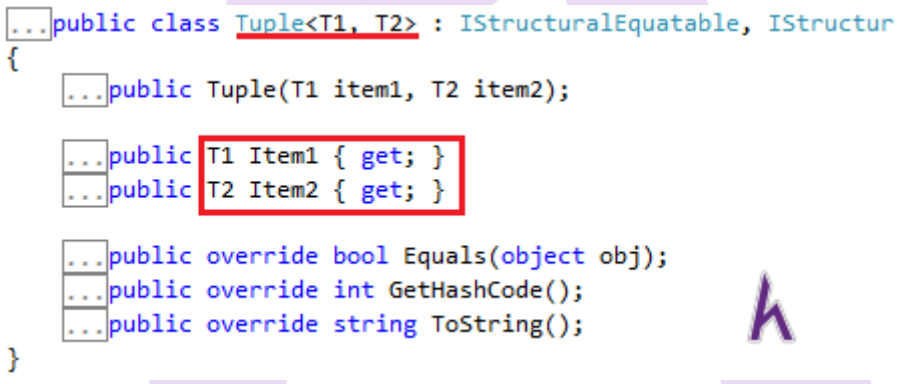
Tuple là một kiểu dữ liệu có cấu trúc, giúp lưu trữ các dữ liệu phức tạp mà không cần phải tạo ra một struct hay class mới (Nếu bạn nào chưa nắm về kiểu dữ liệu có cấu trúc cũng như struct, class thì có thể xem lại bài STRUCT TRONG C# và CLASS TRONG C#).

C# cung cấp cho chúng ta:

- 8 lớp **generic** (khái niệm về generic đã được trình bày trong bài GENERIC TRONG C#) Tuple<>.

```
public class Tuple <T1>

public class Tuple <T1, T2>

public class Tuple <T1, T2, T3>

public class Tuple <T1, T2, T3, T4>

public class Tuple <T1, T2, T3, T4, T5>

public class Tuple <T1, T2, T3, T4, T5, T6>

public class Tuple <T1, T2, T3, T4, T5, T6, T7>

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

Mỗi lớp Tuple<> đã được định nghĩa sẵn các Property có tên Item1, Item2, Item3,... tương ứng với các kiểu dữ liệu T1, T2, T3,... được truyền vào. Hình ảnh sau đây là một ví dụ:



Đây là hình ảnh lớp Tuple <**T1**, **T2**>, khi ta truyền 2 kiểu dữ liệu vào **T1**, **T2** thì trong lớp này sẽ có 2 Property Item1, Item2 có kiểu dữ liệu tương ứng.

- Một **static class** (khái niệm về static class đã được trình bày trong bài STATIC TRONG C#) Tuple. Trong lớp sẽ chứa các hàm Create<> nhằm khởi tạo một Tuple:

```
public static class Tuple
{
    public static Tuple<T1> Create<T1>(T1 item1);
    public static Tuple<T1, T2> Create<T1, T2>(T1 item1, T2 item2);
    public static Tuple<T1, T2, T3> Create<T1, T2, T3>(T1 item1, T2 item2, T3 item3);
    public static Tuple<T1, T2, T3, T4> Create<T1, T2, T3, T4>(T1 item1, T2 item2, T3 item3, T4 item4);
    public static Tuple<T1, T2, T3, T4, T5> Create<T1, T2, T3, T4, T5>(T1 item1, T2 item2, T3 item3, T4 item4, T5
item5);
    public static Tuple<T1, T2, T3, T4, T5, T6> Create<T1, T2, T3, T4, T5, T6>(T1 item1, T2 item2, T3 item3, T4
item4, T5 item5, T6 item6);
    public static Tuple<T1, T2, T3, T4, T5, T6, T7> Create<T1, T2, T3, T4, T5, T6, T7>(T1 item1, T2 item2, T3
item3, T4 item4, T5 item5, T6 item6, T7 item7);
    public static Tuple<T1, T2, T3, T4, T5, T6, T7, Tuple<T8>> Create<T1, T2, T3, T4, T5, T6, T7, T8>(T1 item1, T2
 item2, T3 item3, T4 item4, T5 item5, T6 item6, T7 item7, T8 item8);
}
}
```

Nhìn dài dòng vậy thôi chứ thật ra mỗi hàm Create sẽ tương ứng tạo 1 đối tượng kiểu generic Tuple mình đã giới thiệu ở trên.

Đến đây nhiều bạn sẽ thắc mắc **Tuple được sử dụng khi nào trong khi đã có struct và class?**

- Trường hợp đầu tiên là khi bạn viết một phương thức và muốn trả về 1 lúc nhiều giá trị. Lúc này Tuple sẽ giúp bạn dễ dàng giải quyết mà **không** cần phải tạo thêm struct hay class.
- Trong nhiều trường hợp khác bạn chỉ muốn tạo nhanh 1 đối tượng với 1 vài thuộc tính và chỉ sử dụng 1 lần thôi thì việc dùng struct hoặc class là rất lãng phí, làm chương trình của bạn trở nên dài dòng hơn. Khi đó Tuple được sử dụng như một phương án thay thế tốt hơn vì có sẵn rồi giờ lấy ra dùng thôi không cần khai báo gì nữa.
- Ngoài ra, Tuple đã override sẵn:
    - Phương thức **Equals** (phương thức dùng để so sánh 2 đối tượng).
    - Phương thức **ToString** (Phương thức chuyển giá trị đối tượng sang chuỗi).
    - Phương thức **GetHashCode** (Phương thức trả về mã băm của một đối tượng, dùng để hỗ trợ so sánh 2 đối tượng).

Từ đó chúng ta chỉ việc sử dụng mà không cần phải viết lại những phương thức này.

Cách sử dụng Tuple

Đầu tiên là khởi tạo một đối tượng Tuple chúng ta có 2 cách:

Cách 1: Thông qua phương thức Create trong lớp Tuple:

```
// Khởi tạo Tuple thông qua phương thức Create
var MyTuple = Tuple.Create<int, string>(1, "HowKteam");
```

Ví dụ trên có nghĩa là tạo ra 1 đối tượng (Tuple) có 2 thuộc tính bên trong:

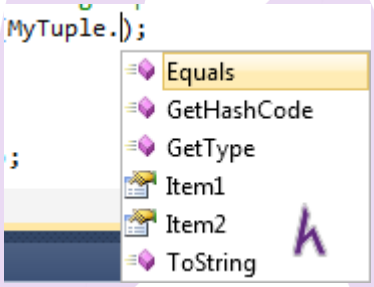
- Thuộc tính **thứ nhất** (Item1) có kiểu dữ liệu là int và khởi tạo giá trị là **1**.
- Thuộc tính **thứ hai** (Item2) có kiểu dữ liệu là string và khởi tạo giá trị là "**HowKteam**".

Cách 2: Thông qua Constructor của các lớp Generic:

```
// Khởi tạo Tuple thông qua constructor của các lớp generic
var MyTuple2 = new Tuple<int, string>(2, "Kteam");
```

Ví dụ này cũng được hiểu tương tự như trên.

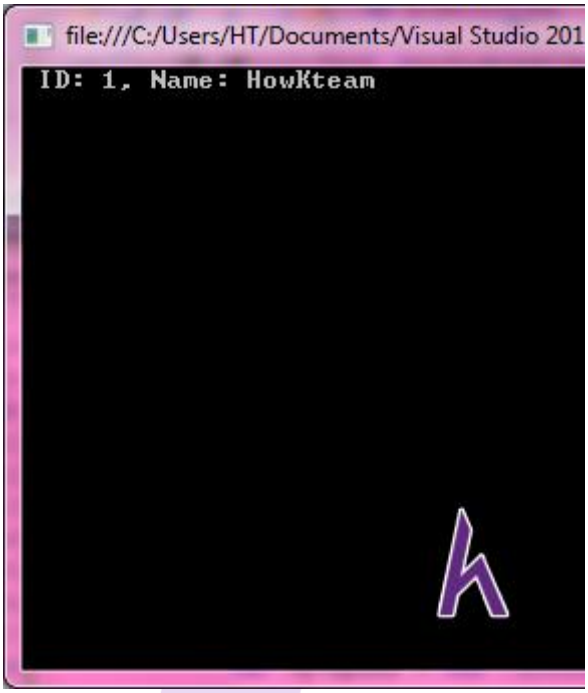
- Khi bạn khởi tạo Tuple có bao nhiêu kiểu dữ liệu thì sẽ có bấy nhiêu thuộc tính **Item** tương ứng. Hình dưới đây là minh hoạ cho Tuple<**T1**, **T2**>:



Ví dụ: in ra giá trị các thuộc tính đã khởi tạo ở trên:

```
// Lấy giá trị bên trong Tuple
Console.WriteLine(" ID: {0}, Name: {1}",MyTuple.Item1, MyTuple.Item2);
```

Kết quả khi chạy chương trình là:



Lúc này ta đã hiểu Tuple cũng chỉ là một lớp bình thường đã được định nghĩa sẵn thôi. Vì vậy để giải quyết bài toán trả về nhiều giá trị cùng một lúc ta đơn giản chỉ cần khai báo kiểu trả về là 1 Tuple mà thôi.

Xét ví dụ sau: Viết chương trình trả về 3 giá trị là ngày, tháng, năm hiện tại của hệ thống.

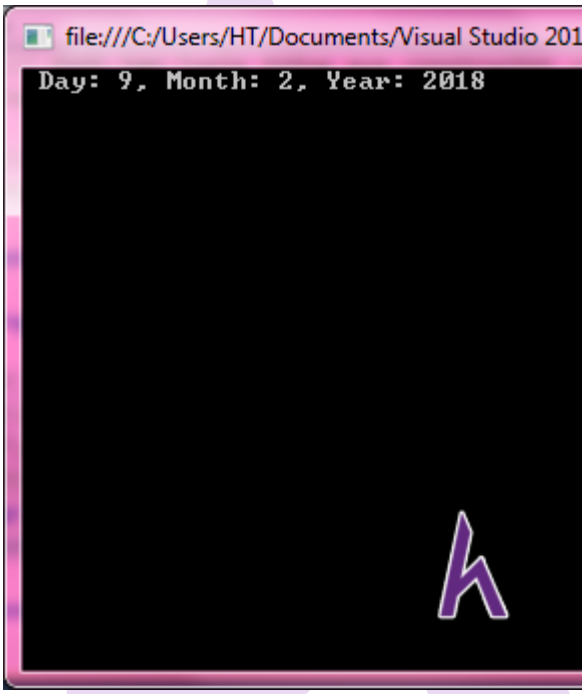
- Đầu tiên mình sẽ viết phương thức trả về 3 giá trị ngày, tháng, năm:

```
/// <summary>
/// Phương thức trả về 1 Tuple có 3 thuộc tính (cả 3 đều có kiểu dữ liệu là int)
/// </summary>
/// <returns></returns>

static Tuple<int, int, int> GetCurrentDayMonthYear()
{
DateTime now = DateTime.Now; // lấy ngày giờ hiện tại của hệ thống.
/* Sử dụng Constructor của Tuple<> để trả về hoặc có thể sử dụng phương thức
Create đã trình bày ở trên. */
return new Tuple<int, int, int>(now.Day, now.Month, now.Year);
}
```

- Trong chương trình chính ta gọi và sử dụng như sau:

```
/*
 * Mình dùng var để C# tự nhận diện kiểu dữ liệu.
 * Bạn có thể khai báo tường minh kiểu dữ liệu là Tuple<int, int, int>
 */

var now = GetCurrentDayMonthYear();
Console.WriteLine(" Day: {0}, Month: {1}, Year: {2}", now.Item1, now.Item2,
now.Item3);
```

- Kết quả khi chạy chương trình là:



Các bạn có thể thử không gọi Item1, Item2, Item3 mà sử dụng phương thức ToString() để xem kết quả thế nào nhé!

> **Lưu ý:** Các thuộc tính Item1, Item2, Item3,... là các thuộc tính chỉ đọc nghĩa là bạn chỉ có thể gọi ra để lấy giá trị chứ không thể gán giá trị cho chúng được.

Kết luận

Nội dung bài này giúp các bạn nắm được:

- Tuple là gì?
- Cách sử dụng Tuple.

Bài học sau chúng ta sẽ cùng tìm hiểu về ICOLLECTION TRONG C#.

Cảm ơn các bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên "**Luyện tập – Thử thách – Không ngại khó**".