Bài: Generic trong C#

Xem bài học trên website để ủng hộ Kteam: Generic 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 BITARRAY TRONG C#. Hôm nay chúng ta sẽ cùng tìm hiểu về **Generic 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:

- Generic là gì?
- Một số loại Generic Collections thông dụng.

Generic là gì?

Nếu bạn đã từng học LẬP TRÌNH C++ thì chắc hẳn bạn đã từng biết tới khái niệm **Template** (hay còn gọi là mẫu).

Template được dùng để tạo ra các lớp, các hàm mà không cần quan tâm đến đối số kiểu dữ liệu là gì. Template được đưa ra với mục đích tăng tính tái sử dụng lại mã nguồn.

Tượng tự Template của C++, **Generic** trong C# cho phép bạn định nghĩa một hàm, một lớp mà không cần chỉ ra đối số kiểu dữ liệu là gì. Tuỳ vào kiểu dữ liệu mà người dùng truyền vào thì nó sẽ hoạt động theo kiểu dữ liệu đó.

Ví dụ: bạn muốn hàm hoán đổi giá trị 2 số nguyên ta sẽ viết như sau:

C#:

```
public static void Swap(ref int a, ref int b)
{
    int temp = a;
    a = b;
    b = temp;
}
```

Mọi thứ đều hoạt động tốt cho đến khi bạn hoán đổi 2 số thực. Khi đó bạn phải viết lại 1 hàm **Swap** mới với kiểu dữ liệu của tham số truyền vào là kiểu số thực.

Mặc dù thao tác giống nhau nhưng ta phải viết hàm này 2 lần. Chính vì vậy mà Generic ra đời để giúp chúng ta giảm thiểu việc code và tăng tính tái sử dụng.

Nếu sử dụng Generic ta sẽ viết như sau:

C#:

```
public static void Swap<T>(ref T a, ref T b)
{
    T temp = a;
    a = b;
    b = temp;
}
```

Bạn chỉ cần đặt 1 chữ cái nào đó thay cho kiểu dữ liệu và khi gọi hàm bạn chỉ ra kiểu dữ liệu bạn đang thao tác là gì. Ví dụ:

C#:

```
int a = 5, b = 7;
double c = 1.2, d = 5.6;

Swap<int>(ref a, ref b);
Swap<double>(ref c, ref d);
```

Khi bạn gọi cú pháp bên dưới thì hàm Swap sẽ chạy và thay ký tự **T** thành kiểu dữ liệu int tương ứng.

:

```
Swap<int>(ref a, ref b)
```

Phía trên mình vừa giới thiệu về Generic cho phương thức. Tiếp theo là Generic cho lớp cũng tương tự.

Ví dụ:

C#:

```
public class MyGeneric<T>
{
    private T[] items;

    public T[] Items
    {
        get { return items; }
    }

    public MyGeneric(int Size)
    {
        items = new T[Size];
    }

    public T GetByIndex(int Index)
    {
        // Nếu index vượt ra khỏi chỉ số phần tử của mảng thì ném ra ngoại lệ
        if (Index < 0 || Index >= items.Length)
        {
            throw new IndexOutOfRangeException();
        }
        else
        {
            return items[Index];
        }
    }

    public void SetItemValue(int Index, T Value)
    {
        if (Index < 0 || Index >= items.Length)
        {
            throw new IndexOutOfRangeException();
        }
        else
        {
            items[Index] = Value;
        }
    }
}
```

Bạn cứ tưởng tượng bình thường bạn viết một lớp cho kiểu int, long, double gì đó thì giờ thay nó bằng **T** còn lại thao tác như bình thường.

Đến khi sử dụng thì ta truyền kiểu dữ liệu thích hợp vào. Ví dụ:

C#:

```
// Khởi tạo 1 mảng số nguyên kiểu int có 5 phần tử
MyGeneric<int> MyG = new MyGeneric<int>(5);

MyG.SetItemValue(0, 10);
```

Khi bạn khai báo cú pháp bên dưới thì trình biên dịch sẽ hiểu **T** trong lớp MyGeneric là kiểu int và thay thế toàn bộ chữ cái **T** trong lớp thành int sau đó thực thi.

:

```
MyGeneric<int>
```

Đặc điểm của Generic

Giúp định nghĩa một thao tác dữ liệu với kiểu dữ liệu chung nhất nhìn hạn chế viết code và tái sử dụng.

Ứng dụng phổ biến nhất của Generic là tạo ra các Generic Collections.

- Ở những bài học trước ta đã tìm hiểu các Collections phổ biến thì nếu các bạn để ý giá trị lưu trữ bên trong đều là object.
- Điều này gây rất nhiều khó khăn nếu như ta muốn quản lý 1 danh sách có cùng kiểu. Vì object có thể chứa được mọi kiểu dữ liệu nên ta khó kiểm soát rằng việc thêm phần tử có phải cùng kiểu dữ liệu ta mong muốn hay không.
- Từ đó Generic Collections ra đời để giúp ta vừa có thể sử dụng được các Collections vừa có thể hạn chế lỗi xảy ra trong quá trình thực thi.

Ngoài ra, Generic còn giúp hạn chế truy cập nếu như không truyền đúng kiểu dữ liệu. Điều này sẽ được trình bày trong những bài học sau.

Trong series này mình sẽ không đào sâu về cách viết Generic mà thay vào đó sẽ giới thiệu cho các bạn một số Generic Collections phổ biến cũng như cách sử dụng chúng.

Một số loại Generic Collections thông dụng

Các Generic Collections đều được xây dựng bắt nguồn từ 1 Collections nào đó có sẵn. Vì thế với mỗi Collections đã học sẽ có một Generic tương ứng.

Một số Generic Collections được sử dụng phổ biến:

LỚP	MÔ TẢ
List<T>	Là một Collections giúp lưu trữ các phần tử liên tiếp (giống mảng) nhưng có khả năng tự mở rộng kích thước. Generic Collections này là sự thay thế cho ArrayList đã học.
Dictionary<Tkey, TValue>	Lớp lưu trữ dữ liệu dưới dạng cặp **Key – Value**. Khi đó ta sẽ truy xuất các phần tử trong danh sách này thông qua Key (thay vì thông qua chỉ số phần tử như mảng bình thường). Generic Collections này là sự thay thế cho Hashtable đã học.
SortedDictionary<Tkey, TValue>	Là sự kết hợp của List và Dictionary. Tức là dữ liệu sẽ lưu dưới dạng **Key – Value**. Ta có thể truy xuất các phần tử trong danh sách thông qua **Key** hoặc thông qua chỉ số phần tử. Đặc biệt là các phần tử trong danh sách này luôn được sắp xếp theo giá trị của **Key**. Generic Collections này là sự thay thế cho SortedList đã học.
Stack<T>	Lớp cho phép lưu trữ và thao tác dữ liệu theo cấu trúc **LIFO (Last In First Out)**. Generic Collections này là sự thay thế cho Stack đã học.
Queue<T>	Lớp cho phép lưu trữ và thao tác dữ liệu theo cấu trúc **FIFO (First In First Out)**. Generic Collections là sự thay thế cho Queue đã học.

Kết luận

Nội dung bài này giúp các bạn nắm được:

- Generic là gì?
- Một số loại Generic Collections thông dụng.

Bài học sau chúng ta sẽ cùng tìm hiểu về LIST 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**".