GIỚI THIỆU VỀ GENERIC TRONG .NET 2.0
07/11/2007 08:30 AM

Generic là một tính năng mới khá thú vị của .net 2.0. Vậy Generic là gì? sử dụng chúng như thế nào? Tại bài viết này chúng ta sẽ trả lời những câu hỏi đó.

Type Safety

Các ngôn ngữ .net (C#, C++, VB..) đều là những ngôn ngữ được định kiểu mạnh, có nghĩa là bạn phải khai báo biến rõ ràng trước khi sử dụng.

Tuy nhiên khi ta sử dụng các kiểu tập hợp (collection) lại không hỗ trợ kiểu an toan(Type safety). Ví dụ kiểu ArrayList, chúng cho phép ta lưu trữ tất cả các object bên trong chúng.

```
using System;
using System.Collections;

namespace TestApp
{
    class Test
    {
        [STAThread]
        static void Main(string[] args)
        {
            ArrayList list = new ArrayList();

            list.Add(3);
```

```
            list.Add(4);
            //list.Add(5.0);

            int total = 0;
            foreach(int val in list)
            {
                total = total + val;
            }

            Console.WriteLine( "Total is {0}", total);
        }
    }
}
```

Khi ta chạy chương trình trên sẽ cho ta kết quả là 7. Nhưng khi ta thêm dòng lệnh

list.Add(5.0);

Chương trình sẽ thông báo lỗi trong thời gian chạy.

Generic là gì?

Generic cho phép chúng ta định kiểu an toàn (type safety). Chúng cho phép ta tạo ra một cấu trúc dữ liệu mà không cần phải xác định đó là kiểu dữ liệu gì. Tuy nhiên khi cấu trúc dữ liệu này được sử dụng, trình biên dịch phải đảm bảo rằng kiểu dữ liệu được sử dụng với nó là kiểu an toàn. Generic cũng tương đương vơi Template trong C++ tuy nhiên việc sử dụng Generic trong .net dễ dàng hơn nhiều so với Template.

```
List<int> aList = new List<int>();
aList.Add(3);
aList.Add(4);
// aList.Add(5.0);

int total = 0;
foreach(int val in aList)
{
   total = total + val;
}
Console.WriteLine("Total is {0}", total);
```

Ta sẽ không thể thực hiện đượng dòng lệnh aList.Add(5.0). Trình biên dịch sẽ thông báo lỗi.

Generic Class

Dưới đây tôi xin giới thiệu cách khai báo cũng như sử dụng các lớp Generic

Kiểu tham số

Nhìn chung Generic class chỉ là một lớp mà nó tiếp nhận các tham số. Chính vì vậy mà các tham số của Generic class chỉ là những loại trừu tượng mà nó được sử dụng trong cấu trúc của một hay nhiều kiểu xác định trong thời gian chạy. Thuật ngữ này làm ta liên tưởng đến các kiểu tham số của Generic Class có thể đáp ứng như một PlaceHolders mà được thay thế bởi kiểu dữ liệu thực khi một Generic class được xây dựng.

```
public class Stack<T> {
    private T[] _items;
    private int _count;
    public void Push(T item) {...}
    public T Pop() {...}
}
```

Overload Type

```
public class MyType {
}
public class MyType<T> {
...
}
public class MyType<T, U> {
...
}
```

Thừa kế

```
public class MyBaseClass<U> {
    private U _parentData;
    public MyBaseClass() {}
    public MyBaseClass(U val) {
        this._parentData = val;
    }
}

public class MySubClass<T, U> : MyBaseClass<U> {
    private T _myData;
```

```
    public MySubClass() {}
    public MySubClass(T val1, U val2) : base(val2) {
        this._myData = val1;
    }
}
```

Rằng buộc tham số

Generic cho ta viết một lớp mà không cần xác định kiểu dữ liệu cụ thể, nhưng vẫn cho phép người sử dụng lớp đó chỉ ra kiểu dữ liệu cụ thể sẽ sử dụng. Điều này tạo ra sự linh hoạt bằng cách thay thế một số rằng buộc về kiểu mà có thể được sử dụng trong các tham số

```
public static T Max<T>(T op1, T op2) where T : IComparable
{
    if (op1.CompareTo(op2) < 0)
    return op1;
    return op2;
}
```

Tong ví dụ tôi đã chỉ ra rằng buộc mà kiểu được sử dụng cho kiểu tham số phải thực thi giao diện IComparable.

Các kiểu rằng buộc có thể sử dụng.

where T : struct
where T : class
where T : new() hàm khởi tạo không tham số
where T : class_name kiểu lớp mà tham số phải thừa kế
where T : interface_name kiểu giao diện mà tham số phải thực thi.

Cũng có thể sử dụng kết hợp nhiều rằng buộc như where T : IComparable, new().

Kết thúc

Trên đây là những giới thiệu cơ bản nhất về Generic trong .net 2.0. Chúc bác bạn thành công.