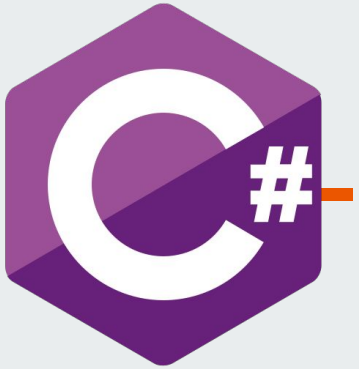

Bài 11

Collection Generics

Nội dung

- Giới thiệu về Generics
- Tạo Generic class và sử dụng
- Tạo Generic method và sử dụng
- Tạo Generic interface và sử dụng
- Giới thiệu một số Generic Collections trong .NET
- Tìm hiểu một số Generic Collection (List<>, Dictionary<>, SortedList<>)

Giới thiệu về Generics

- Generic được giới thiệu từ .NET Framework 2.0.
- Generics có thể khai báo như là một tham số khi định nghĩa các class, method, interface, structure, delegate.
- Generics khai báo theo mẫu Type<T,…>
- Generics có thể sử dụng với bất kỳ kiểu gì.
- Generics đảm bảo tính an toàn về kiểu khi thao tác.
- Generics giúp chương trình chạy nhanh do không phải chuyển đổi kiểu trong quá trình sử dụng.

Tạo lớp Generic

- Cú pháp

<access_modififer> class <ClassName><<type parameter list>> [where <type parameter constraint clause>]

```
//Định nghĩa lớp Generic
class General<T>
{
    T[] a;
    int count;
    public General(int n)
    {
        a = new T[n];
        count = 0;
    }
    public void Add(T value)
    {
        if (count < a.Length)
        {
            a[count] = value;
            count++;
        }
    }
    public void Display()
    {
        foreach (var v in a)
        {
            Console.WriteLine(v);
        }
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        //Sử dụng lớp generic với kiểu nguyên
        General<int> general = new General<int>(3);
        general.Add(5);
        general.Add(7);
        general.Add(2);
        general.Display();
        //Sử dụng lớp generic với kiểu chuỗi
        General<string> names = new General<string>(3);
        names.Add("Hoa");
        names.Add("Dung");
        names.Add("Hanh");
        names.Display();
    }
}
```

Phương thức Generic

- Phương thức generic là các phương thức được khai báo một kiểu chung có dạng **MethodName<T>([T p1, Tp2,..])** cho các tham số của nó mà không chỉ ra kiểu cụ thể. Khi gọi phương thức chúng ta mới chỉ ra kiểu cụ thể cho nó.

```
class Utility
{
    //Định nghĩa phương thức generic
    public void Swap<T>(ref T a, ref T b)
    {
        T tg = a;
        a = b;
        b = tg;
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        //sử dụng phương thức generic
        Utility u = new Utility();
        int x = 30, y = 5;
        //áp dụng với kiểu số
        u.Swap<int>(ref x, ref y); //bỏ <int> cũng được nhé
        //áp dụng với kiểu chuỗi
        string st1 = "CHUNGLD", st2 = "HOANT";
        u.Swap<string>(ref st1, ref st2);//bỏ <string> cũng được nhé
    }
}
```

Generic Interfaces

- Tương tự như class chúng ta có thể định nghĩa Generic cho các interface
- Cú pháp

```
<access_modifier> interface <InterfaceName><<type parameter list>> [where
<type parameter constraint clause>]
```

```
//Tạo generic interface
public interface IMath<T>
{
    T Add(T a, T b);
    T Sub(T a, T b);
}
```

```
//Tạo lớp thực thi generic interface với tham số T là int
class Numbers : IMath<int>
{
    public int Add(int a, int b)
    {
        return a + b;
    }

    public int Sub(int a, int b)
    {
        return a - b;
    }
}
```

Giới thiệu về một số Generic Collection trong .NET

- .NET Framework cung cấp cho chúng ta một số các Generic Collection rất hữu ít.

`System.Collections.Generic`

Cung cấp các Generic collection cho phép lập trình viên customize.

`System.Collections.ObjectModel`

Cung cấp các Generic collection cho phép lập trình viên tạo Dynamic và Read only.

Lớp ReadOnlyCollection

- Nằm trong namespace **System.Collection.ObjectModel**, nó cung cấp một tập dữ liệu chỉ đọc

```
class Program
{
    static void Main(string[] args)
    {
        List<string> countries = new List<string>();
        countries.Add("Vietnamese");
        countries.Add("France");
        countries.Add("Canada");
        countries.Add("India");
        ReadOnlyCollection<string> countriesReadOnly =
            new ReadOnlyCollection<string>(countries);
        Console.WriteLine("Values stored in the read only collection");
        foreach (string str in countriesReadOnly)
        {
            Console.WriteLine(str);
        }
    }
}
```

Một số lớp Generic trong System.Collection.Generics

- Bảng sau liệt kê danh sách các lớp Collection và Generic Collections tương ứng

Collections	Generic Collections
ArrayList	List<>
Hashtable	Dictionary<>
SortedList	SortedList<>
DictionaryEntry	KeyValuePair<>

Lớp List<>

- Lớp List< >: tương tự ArrayList nhưng khi tạo chúng ta cần chỉ ra kiểu cả các phần tử trong danh sách.

```
class Program
{
    static void Main(string[] args)
    {
        //tạo tập hợp chỉ chứa kiểu số nguyên
        List<int> numbers = new List<int>();
        numbers.Add(10);
        numbers.Add(3);
        numbers.Add(5);
        //numbers.Add("w3fair.com");//dòng này biên dịch sẽ báo lỗi nhé
        foreach (int n in numbers)
        {
            Console.WriteLine(n);
        }
    }
}
```

Lớp Dictionary<,>

- Lớp Dictionary<,>: tương tự lớp Hashtable nhưng key và value phải chỉ ra kiểu khi tạo.

```
class Program
{
    static void Main(string[] args)
    {
        Dictionary<int, string> dep = new Dictionary<int, string>();
        dep.Add(101, "Accounting");
        dep.Add(102, "Human Resource");
        dep.Add(103, "Infomation Technology");
        dep.Add(104, "System");
        foreach (var key in dep.Keys)
        {
            Console.WriteLine(key+":"+ dep[key]);
        }
    }
}
```

Lớp StoredList<,>

- Lớp StoredList< >: tương tự lớp Dictionary< > nhưng các phần tử được sắp xếp theo key.

```
class Program
{
    static void Main(string[] args)
    {
        SortedList<string, int> numbers = new SortedList<string, int>();
        numbers.Add("Three", 3);
        numbers["One"] = 1;
        numbers.Add("Two", 2);
        numbers.Add("Four", 4);
        numbers.Add("Ten", 10);
        //in ra danh sách sắp xếp theo key
        foreach (string key in numbers.Keys)
        {
            Console.WriteLine(key + ":" + numbers[key]);
        }
    }
}
```

Khởi tạo nhanh collection 1-2

- Từ C# 3.0 trở lên, .NET cung cấp cách khởi tạo nhanh một tập hợp mà không cần sử dụng phương thức Add/AddRange như phiên bản trước.

```
//cách khởi tạo cũ
List<string> animal = new List<string>();
animal.Add("Tung");
animal.Add("Cuc");
animal.Add("Truc");
animal.Add("Mai");
//cách khởi tạo mới
List<string> animal1 = new List<string>() { "Tung", "Cuc", "Truc", "Mai" };
```

```
//Khởi tạo nhanh Dictionary
Dictionary<string, int> flowers = new Dictionary<string, int>()
{
    {"Hoa Lan",20000},{"Hoa Hong",24000},{"Hoa Ly",23000}
};
```

Khởi tạo nhanh collection 2-2

```
class Employee
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Address { get; set; }
}
```

```
//Khởi tạo nhanh với dữ liệu dạng class
List<Employee> list = new List<Employee>()
{
    new Employee{Id=1,Name="Tran Manh",Address="Ha Noi"},
    new Employee{Id=2,Name="Tri Dung",Address="Ha Noi"},
    new Employee{Id=3,Name="Thien Phi",Address="Ha Giang"}
};
```

HỎI ĐÁP

TRẢI NGHIỆM THỰC HÀNH