什么是泛型(Generic)

泛型是CLR和编程语言提供的特殊机制, 它支持另一种形式的代码重用,即 “算法重用”.

简单的说就是: 在算法定义的时候并不知道所操作的类型, 在使用的时候才确定操作类型.

因为这个我们所操作的类型在定义时不知道, 所以我们需要定义一个或者多个类型参数以供使用时传入. 类型参数一般在<> 中间定义, 比如: <T>, <TKey, TValue>

泛型的优势: (示例: GenericAdvantage)

1. 类型安全:
2. 更加清晰的代码.
3. 更佳的性能.
4. 源代码保护.

CLR类库中提供的一些泛型: (示例: CLRGeneric).

1. 泛型类型: List<T>, Comparer<T>, Dictionary<TKey, TValue> 等.
2. 泛型接口: IList<T>, IEnumerable<T>, ICollection<T>, IDictionary<TKey, TValue>等.
3. 泛型方法: Array.Sort<T>(), Array.BinarySearch<T>(). 等.
4. 泛型代理: Action<T>, Predicate<T>, Fun<T, TResult> 等.

泛型的基本结构.

泛型类型:

泛型类型的定义, 继承和使用. (示例: GenericClass).

|  |
| --- |
| public class Node<T>  public class IntNode : Node<int>  Node<char> head = new Node<char>('C'); |

Note: 有的时候泛型会将开发人员弄糊涂. 因为代码中可能散布着大量的”<>”, 影响可读性, 有的时候大家会定义一个新的非泛型类型,它是一个派生, 并制定了所有的类型实参. 也可以用Using来实现.

开放类型和封闭类型. (示例: OpenCloseGenericClass).

开放类型: 具有泛型类型参数的类型称开放类型. CLR禁止构造开发类型的任何实例.

封闭类型: 为所有类型参数传递的都是实际数据类型.

泛型接口:

定义泛型接口.

public interface INode<T>

泛型委托:

定义泛型委托:

public delegate TResult CallMe<in T, out TResult>(T arg)

Note 建议尽量使用在FCL中预定义的泛型类型Action, Predicate 和Func委托.

泛型方法: (示例: GenericMethod)

泛型方法的定义.

类型推断: 编译器会在调用一个泛型方法时自动判断(或者推断)要使用的类型.

泛型的可验证性和约束.

约束: 约束的作用是限制能制定成泛型实参的类型数量, 通过限制类型的数量, 我们可以对那些类型执行更多的操作.

主要约束: (示例: PrimaryConstraint).

类型参数可以指定零个或者一个主要约束, 主要约束可以是一个应用类型.

次要约束: (示例: SecondaryConstraint).

类型参数可以指定零个或者多个次要约束, 次要约束有2种.

接口约束: 指定一个接口约束时, 向编译器承诺一个指定的类型实参是实现了接口的一个类型. 因为支持指定多个接口约束, 所以类型实参必须实现所有的接口.

类型参数约束: 是在指定的类型实参直接, 必须存在一个关系. 一个类型实参可以指定零个或者多个参数约束.

构造器约束:

一个类型参数可以指定零个或者一个构造器约束. 指定构造器约束相当于向编译器承诺一个指定的类型实参是实现了公共无参的一个非抽象类型.

泛型类型变量设为默认值. Default(T).