C# 泛型的协变和逆变 - 文章 - 伯乐在线

1. 可变性的类型：协变性和逆变性

可变性是以一种类型安全的方式，将一个对象当做另一个对象来使用。如果不能将一个类型替换为另一个类型，那么这个类型就称之为：不变量。协变和逆变是两个相互对立的概念：

- 如果某个返回的类型可以由其派生类型替换，那么这个类型就是支持协变的
- 如果某个参数类型可以由其基类替换，那么这个类型就是支持逆变的。

2. C# 4.0对泛型可变性的支持

在C# 4.0之前，所有的泛型类型都是不变量——即不支持将一个泛型类型替换为另一个泛型类型，即使它们之间拥有继承关系，简而言之，在C# 4.0之前的泛型都是不支持协变和逆变的。

> C# 4.0通过两个关键字：out和in来分别支持以协变和逆变的方式使用泛型。

我们来看一段利用了协变类型参数的代码：

```
C#
public class BaseClass
{
//...
}
public class DerivedClass : BaseClass
{
//...
}
```

下面我们利用协变类型参数，可以执行类似于普通的多态性的分配：

```
C#
IEnumerable<DerivedClass> d = new List<DerivedClass>();
IEnumerable<BaseClass> b = d;
```

在上面的实例中，在C# 4.0之前是不能正常编译的，除了对赋值给基类集合时将子类集合做一个强制转换，但是在运行时仍然会抛出一个类型转换的异常。

下面我们再看一个关于逆变的实例代码：

```
C#
Action<BaseClass> b = (target) => { Console.WriteLine(target.GetType().Name); };
Action<DerivedClass> d = b;
```

```
d(new DerivedClass());
```

在上面的示例中我们 Action<BaseClass> 类型的委托分配给类型 Action<DerivedClass> 的变量，根据逆变的定义我们可以知道 Action<T> 类型是支持逆变的。

为什么IEnumerable<T> 和 Action<T> 可以分别支持类型的逆变和协变呢？我们查看这两个类型在 .NET 中的定义：

```
C#//IEnumerable<T> 接口的定义(支持协变)
public interface IEnumerable<out T> : IEnumerable
//Action<T> 委托的定义(支持逆变)
public delegate void Action<in T>(T obj);
```

为了保证类型的安全，C#编译器对使用了 out 和 in 关键字的泛型参数添加了一些限制：

支持协变(out)的类型参数只能用在输出位置：函数返回值、属性的get访问器以及委托参数的某些位置
支持逆变(in)的类型参数只能用在输入位置：方法参数或委托参数的某些位置中出现。

3. C#中泛型可变性的限制

1. 不支持类的类型参数的可变性

只有接口和委托可以拥有可变的类型参数。in 和 out 修饰符只能用来修饰泛型接口和泛型委托。

2. 可变性只支持引用转换

可变性只能用于引用类型，禁止任何值类型和用户定义的转换，如下面的转换是无效的：

- 将 IEnumerable<int> 转换为 IEnumerable<object> ——装箱转换
- 将 IEnumerable<short> 转换为 IEnumerable<int> ——值类型转换
- 将 IEnumerable<string> 转换为 IEnumerable<XName> ——用户定义的转换

3. 类型参数使用了 out 或者 ref 将禁止可变性

对于泛型类型参数来说，如果要将该类型的实参传给使用 out 或者 ref 关键字的方法，便不允许可变性，如：

```
C#
```

这段代码编译器会报错。

4. 可变性必须显式指定

从实现上来说编译器完全可以自己判断哪些泛型参数能够逆变和协变，但实际却没有这么做，这是因为C#的开发团队认为：

> 必须由开发者明确的指定可变性，因为这会促使开发者考虑他们的行为将会带来什么后果，从而思考他们的设计是否合理。

5. 注意破坏性修改

在修改已有代码接口的可变性时，会有破坏当前代码的风险。例如，如果你依赖于不允许可变性的is或as操作符的结果，运行在.NET 4时，代码的行为将有所不同。同样，在某些情况下，因为有了更多可用的选项，重载决策也会选择不同的方法。所以在对已有代码引入可变性时要做好足够的单元测试以及防御措施。

6. 多播委托与可变性不能混用

下面的代码能够通过编译，但是在运行时会抛出 ArgumentException 异常：

C#