

.NET框架2.0新特性综述

卢斌

Software Development Engineer
Microsoft Corporation

BCL 新功能

- Serial port
- Compression
- Strongly typed resources
- Console support
- Threading
- Diagnostics
- Networking
- ...

演示

Strongly Typed Resources

Generics “泛型”

```
public class List<T>
{
    private T[] elements;
    private int count;
    public void Add(T element)
    {
        if (count == elements.Length) Resize(count * 2);
        elements[count++] = element;
    }
    public T this[int index]
    {
        get { return elements[index]; }
        set { elements[index] = value; }
    }
    public List<int> intList = new List<int>();
    public intList.Add(1);           // No boxing
    public intList.Add(2);           // No boxing
    public intList.Add("3");         // Compile-time error
    int i = intList[0];             // No cast required
```

Generics “泛型”

- 为什么要Generics?
 - 编译时类型验证
 - 高性能 (不用装箱box, 不用转换downcasts)
 - 减少代码累赘 (typed collections)
- VB, C#, MC++ 编写和运用 generics
- Generics国际标准化

Generics in VB

```
Public Class List(Of ItemType)
    Private elements() As ItemType
    Private elementCount As Integer
    Public Sub Add(ByVal element As ItemType)
        If elementCount = elements.Length Then
            Resize(elementCount * 2)
        elements(elementCount) = element
        Count += 1
    End Sub
    Public Default Property Item(ByVal index As Integer) As ItemType
        Get
            Return elements(index)
        End Get
        Set (ByVal value As ItemType)
            elements(index) = value
        End Set
    End Property
    Public Shared Sub New()
        Dim intList As New List(Of Integer)
        intList.Add(1)                      ' No boxing
        intList.Add(2)                      ' No boxing
        intList.Add("3")                    ' Compile-time error
        Dim i As Integer = intList(0)       ' No cast required
    End Class
```

Generics in C++

```
generic<typename T>
public ref class List {
    array<T>^ elements;
    int count;
public:
    void Add(T element) {
        if (count == elements->Length) Resize(count * 2);
        elements[count++] = element;
    }
    property T default [int index] {
        T get() { return elements[index]; }
        void set(T value) { elements[index] = value; }
    }
    property int Count {
        int get() { return count; }
    }
};
```

List<int>^ intList = gcnew List<int>();
intList->Add(1); // No boxing
intList->Add(2); // No boxing
intList->Add("3"); // Compile-time error

int i = intList[0]; // No cast required

基础类库泛类 (Generics in BCL)

- System.Collections.Generic classes
 - List<T>
 - Stack<T>
 - Dictionary<K, V>
 - Queue<T>
- System.Collections.Generic interfaces
 - IList<T>
 - IDictionary<K, V>
 - ICollection<T>
 - IEnumerable<T>
 - IEnumerator<T>
 - IComparable<T>
 - IComparer<T>
- Nullable(Of T)
 - Dim intValue as Nullable(Of Integer) = 5
If intValue.HasValue Then ' checks for a value
- EventHandler<T>
 - delegate void EventHandler<T>(Object sender, T e)
where T : EventArgs

提高 CLR 性能

长远目标：CLR性能指标与非托管本机代码一致

- 减少多个托管进程的marginal cost
- 减少托管应用程序起动时间和工作集

NGEN

- 将 IL 编译成本机代码，然后存盘
- 好处：无需反复将 IL 编译成本机代码，类的布局也已定型，起动时间更短
- CLR 2.0: 显著地减少了private, non-shareable 工作集
- ngen install, ngen update, ngen /queue

使.NET框架成为更快的编程平台

提高现有API的性能

- 降低应用程序域(AppDomain)开销, 应用程序域之间的方法调用 (加快 ~1.1倍-50倍)
- 委托(delegate)创建 (~10倍) 和调用 (~2倍)
- UTF8Encoding: translation (~2.5倍)
- 新的高性能 API
- 更快的资源查找API
- Lightweight CodeGen: 只生成关键代码 (与 Reflect Emit 对比)

演示

TryParse

走向 64 位

- 64 位 Windows 服务器和工作站
 - 支持 IA64 和 X64 (AMD64)
 - Windows Server 2003 SP1和未来的64 位 Windows Client
- 支持 WoW64
 - 32位应用程序可在 64位机上运行
 - CLR1.0 和1.1应用程序在 WoW64上运行
- VS: 作为32位应用程序运行
 - 可以开发, 调试和部署32位和64位应用程序
- 新的CLR 64位实时编译, 垃圾回收和程序调试服务



