您的潜力，我们的动力

Microsoft
微软（中国）有限公司

.NET Compact Framework P/Invoke编程

马宁

Windows Mobile MVP

msdn

MSDN Webcasts

Overview

- P/Invoke概述
- DllImportAttribute属性
- Data Marshaling
- 创建托管包装类
- 错误控制
- P/Invoke 在.NET CF与.NET中的不同
- 第三方工具

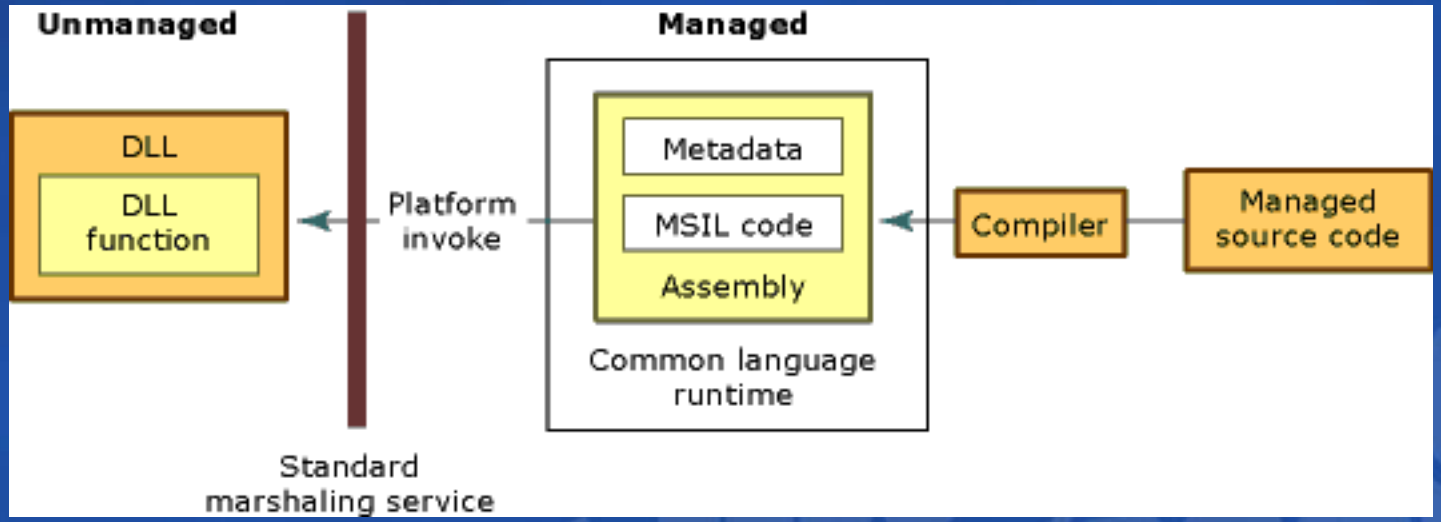
P/Invoke介绍

- Platform Invoke,是允许托管代码调用DLL库的本地代码函数的服务
- P/Invoke定位和调用export函数，排列参数穿越托管边界
- 命名空间 System.Runtime.InteropServices
- 为什么使用P/Invoke?
  - Win32 API包含完整的系统功能调用
  - 调用.NET CF中不具备的功能函数
  - 使用Native代码提高系统效率
  - 调用自定义DLL函数
  - 增强系统的安全特性

P/Invoke使用原则

- 在.NET CF不能提供功能时，使用P/Invoke调用系统函数
- Win32 API与.NET CF错误控制不同
  - Win32 API返回值，GetLastError获得错误值
  - .NET CF抛出异常
- P/Invoke函数以元数据形式存在。在JIT编译时，将函数实现与本地代码连接。
- 进行 P/Invoke 时，不要使应用程序逻辑直接属于任何外部方法或其中的构件。

A Closer Look at P/Invoke



- Locates the DLL containing the function.
- Loads the DLL into memory.
- Locates the address of the function in memory and pushes its arguments onto the stack, marshaling data as required.
- Transfers control to the unmanaged function.

A P/Invoke Sample

- Win32 API原型

```
BOOL MessageBeep(UINT uType);
```

- P/Invoke声明

```
[DllImport("User32.dll")]
static extern Boolean MessageBeep(UInt32 beepType);
```

- C#调用

```
MessageBeep(0);
```

demo

使用P/Invoke拨打电话

DllImportAttribute属性

```
[DllImport("User32.dll")]
static extern Boolean MessageBeep(UInt32 beepType);
```

- DllImport定位API所在的DLL
- DllImportAttribute 属性
  - EntryPoint 指定函数入口点
  - CharSet 字符集，默认值CharSet.Unicode
  - SetLastError 获取Win32错误值，默认值false
  - CallingConvention 函数调用形式，.NET CF只支持默认值CallingConvention.Winapi

Data Marshaling

- 将托管数据类型转换为非托管数据类型，传递给Native函数
- 支持数值类型、字符串类型、结构体、浮点类型
- 引用类型作为指针传递，不支持在引用类型上使用out或ref关键字
- .NET CF中返回值不能超过32bits
- BOOL映射为System.Boolean,在.NET CF中为1-byte integer
- 字符类型映射为System.Char
- Handle类型使用System.IntPtr类型
  - Int32.Zero 用于比较或设置NULL值
  - IntPtr大小根据操作系统确定

数据类型

Win32 Types	Specification	CLR Type
char, INT8, SBYTE, CHAR	8-bit signed integer	System.SByte
short, short int, INT16, SHORT	16-bit signed integer	System.Int16
int, long, long int, INT32, LONG32, BOOL, INT	32-bit signed integer	System.Int32
unsigned, unsigned int, ULONG, DWORD, UINT	32-bit unsigned integer	System.UInt32
unsigned short, UINT16, WORD, ATOM, WCHAR,	16-bit unsigned integer	System.UInt16
unsigned char, UINT8, UCHAR, BYTE	8-bit unsigned integer	System.Byte

指针参数

- 使用ref或out关键字来传递值类型的引用
- Marshal引用类型时，不需要使用ref或out
- out关键字表示参数会被调用函数传递出来
- ref表示数据可以在被调用的函数的内部和外部传递
- out不必被初始化，ref必须被初始化

```
BOOL FileEncryptionStatus(LPCTSTR lpFileName,
                          LPDWORD lpStatus);
```

```
[DllImport("Advapi32.dll", CharSet=CharSet.Auto)]
static extern Boolean FileEncryptionStatus(
                          String filename,
                          out UInt32 status);
```

Marshaling Strings

- .NET CF只支持Unicode字符串
- System.String输入字符串参数
  - 对象分配在托管Heap上，将指针传递给Native函数
- System.StringBuilder输入或输出参数
  - 使用前必须初始化，以分配足够的空间
- LPTSTR和LPCTSTR声明中，C表示字符串为const,不能被函数修改；T表示可能为Unicode或ANSI
- 字符串作为返回值时，传递指针到IntPtr中。使用Marshal.PtrToStringUni转换为string类型

Marshaling Strings Sample

```
// ** Documentation for Win32 GetShortPathName() API Functic
DWORD GetShortPathName(
   LPCTSTR lpszLongPath,      // file for which to get short path
   LPTSTR lpszShortPath,      // short path name (output)
   DWORD cchBuffer            // size of output buffer
);

[DllImport("Kernel32", CharSet = CharSet.Auto)]
static extern Int32 GetShortPathName(
   String path,               // input string
   StringBuilder shortPath,   // output string
   UInt32 shortPathLength);   // StringBuilder.Capacity
```

Marshaling Structures

- struct为值类型，不在托管Heap上分配
- 结构只能包含值类型，不能包括数组或字符串
- 使用struct关键字定义托管结构类型
- 也可以使用class定义结构类型
  - 在.NET CF中StructLayout的默认值为LayoutKind.Sequential，所以不必设置
- .NET CF不支持Native函数返回值是一个指向结构的指针，必须使用Marshal.PtrToStructure手工转换

Structures Sample

```
[StructLayout(LayoutKind.Sequential)]
class SystemPowerStatus
{
    public ACLineStatus _ACLineStatus;
    public BatteryFlag _BatteryFlag;
    public Byte _BatteryLifePercent;
    public Byte _Reserved1;
    public Int32 _BatteryLifeTime;
    public Int32 _BatteryFullLifeTime;
}
```

```
struct SystemPowerStatus
{
    public ACLineStatus _ACLineStatus;
    public BatteryFlag _BatteryFlag;
    public Byte _BatteryLifePercent;
    public Byte _Reserved1;
    public Int32 _BatteryLifeTime;
    public Int32 _BatteryFullLifeTime;
}
```

```
[DllImport("Kernel32", EntryPoint="GetSystemPowerStatus")]
static extern Boolean GetSystemPowerStatusVal(out SystemPowerStatus sps);

[DllImport("Kernel32", EntryPoint="GetSystemPowerStatus")]
static extern Boolean GetSystemPowerStatusRef(SystemPowerStatus sps);
```

Passing Non-Integral Types

- .NET CF中浮点类型不能marshal为值类型
- 可以通过指针传递给Native函数
- 64bit integers也可以通过指针传递
- 可以创建一个Native包装函数，接受托管类型传递的指针，再将浮点数传递给函数

Marshaling Complex Types

- 复杂对象是指包含数组、结构或字符串的结构体
- .NET CF 2.0支持MarshalAsAttribute类型
- C#中使用unsafe和fixed关键字操作内存
- GC过程不回收unsafe方法中的对象
- unsafe允许托管程序直接访问内存

```
fixed(char *  type = string.ToCharArray())
{
}
```

您的潜力，我们的动力

Microsoft
微软（中国）有限公司

demo

P/Invoke传递复杂类型

msdn

MSDN Webcasts

创建托管包装类

- P/Invoke函数作为托管包装类的私有方法
- 可以为每一个API函数创建一个单独的类
- 可以按程序逻辑分类，将API封装在一个类中
- 创建托管包装类，可以重用托管API，而不必更改程序其余代码
- 提供缓冲方法，减少访问冲突和其他低级破坏对应用程序的影响
- 向该方法添加最小的 CLR 样式
  - 将 Windows API 函数返回值的失败转换成CLR 异常
  - 使用托管枚举类型来避免幻数向整个应用程序代码扩散

Sample Wrapper Class


```
namespace Wintellect.Interop.Sound
{
   using System;
   using System.Runtime.InteropServices;
   using System.ComponentModel;

   sealed class Sound
   {
  public static void MessageBeep(BeepTypes type)
      {
         if (!MessageBeep((UInt32)type))
         {
            Int32 err = Marshal.GetLastWin32Error();
            throw new Win32Exception(err);
         }
      }

      [DllImport("User32.dll", SetLastError = true)]
      static extern Boolean MessageBeep(UInt32
beepType);
      private Sound() { }
   }
}
```

```
enum BeepTypes
{
   Simple = -1,
   Ok = 0x00000000,
   IconHand = 0x00000010,
   IconQuestion = 0x00000020,
   IconExclamation = 0x00000030,
   IconAsterisk = 0x00000040
}
```

错误控制

- P/Invoke声明与DLL函数定义不匹配，则抛出NotSupportedException
- DLL函数入口点不存在，抛出MissingMethodException
- 如果Native函数返回错误，我们将通过Marshal.GetLastWin32Error 得到错误值，转换为异常

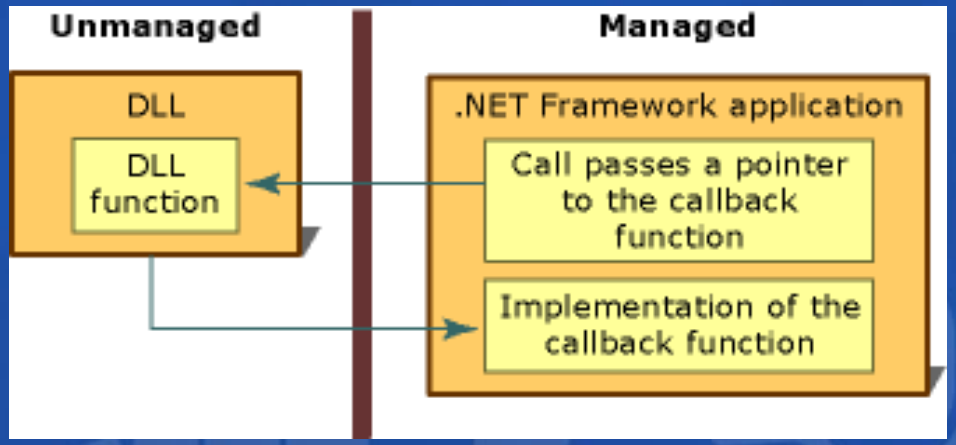
```
if(!MessageBeep((UInt32) type)){
        Int32 err = Marshal.GetLastWin32Error();
        throw new Win32Exception(err);
}
```

.NET CF与.NET的不同

- Unicode支持
  .NET CF只支持Unicode，支持CharSet.Auto和CharSet.Unicode，不支持CharSet.ANSI
- Calling Convention
  .NET CF只支持Winapi（__stdcall）
- .NET CF 2.0开始支持callback函数
- 不同的异常
- 处理Windows Message
  - .NET中Form存在Handle属性，可以重载DefWndProc
  - .NET CF支持MessageWindow and Message

Callback函数

- .NET CF 2.0支持delegates传递给Callback函数
- 通过callback delegates实现Native控件的子类化
- 在Native Callback完成工作之前，Delegates不被GC回收




```
public delegate int EnumDelegate(
  IntPtr hwnd, int LParam);

[DllImport("coredll.dll")]
static extern int EnumWindows(
  EnumDelegate d,
  Int lParm);
```

demo

P/Invoke调用Delegates

OpenNETCF

- OpenNETCF是提高Windows Mobile开发效率的开源类库
- OpenNETCF.Win32命名空间提供了对Windows CE API的封装
- 可以直接将源代码直接加入到自己的工程中
- www.opennetcf.org

demo

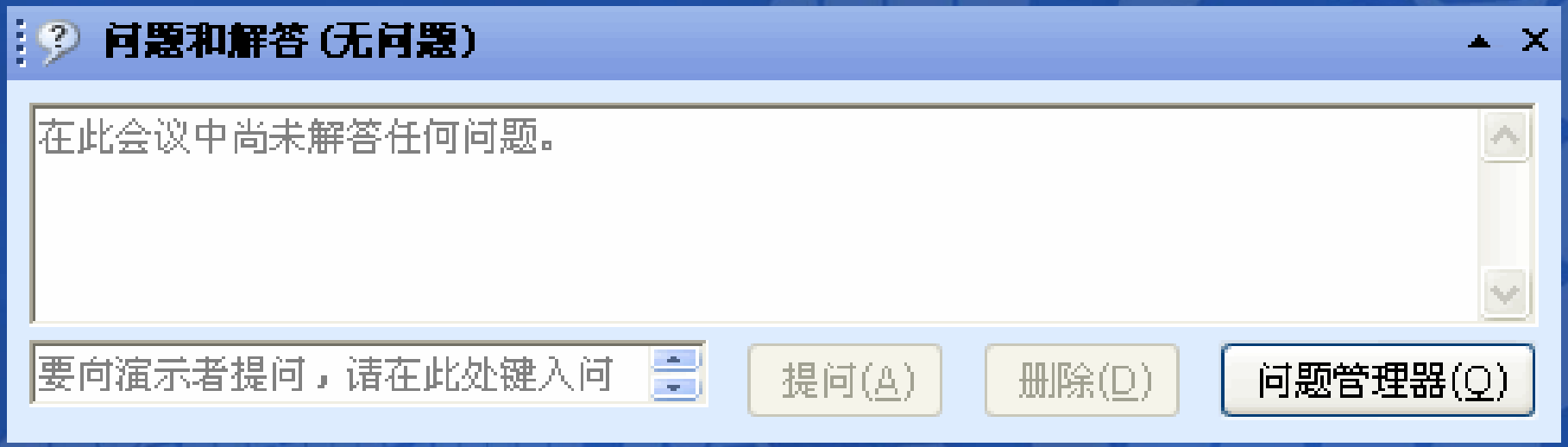
OpenNETCF访问注册表

获取更多MSDN资源

- **MSDN中文网站**
  http://www.microsoft.com/china/msdn
- **MSDN中文网络广播**
  http://www.msdnwebcast.com.cn
- **MSDN Flash**
  http://www.microsoft.com/china/newsletter/case/msdn.aspx
- **MSDN开发中心**
  http://www.microsoft.com/china/msdn/DeveloperCenter/default.mspx

Question & Answer

如需提出问题，请单击“提问”按钮并在随后显示的浮动面板中输入问题内容。一旦完成问题输入后，请单击“提问”按钮。




您的潜力，我们的动力
Microsoft
微软（中国）有限公司
Microsoft
msdn
MSDN Webcasts