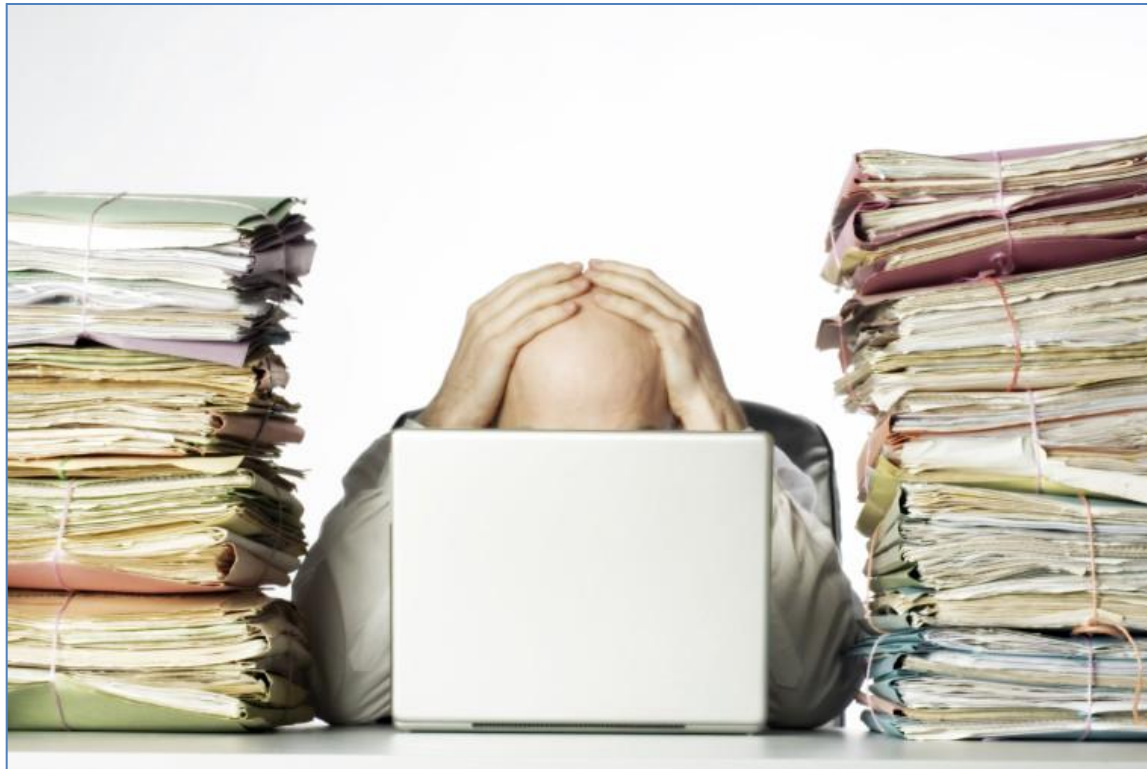


Унаследованный код (legacy code)



- 1. Пространство `System.Runtime.InteropServices`.**
- 2. Взаимодействие с модулями Dll**
 - a. Класс `DllImportAttribute`.**
 - b. Поле `ExactSpelling`.**
 - c. Поле `EntryPoint`.**
 - d. Поле `CharSet`.**
 - e. Поле `CallingConvention`.**
 - f. Поле `SetLastError`.**
- 3. Примеры использования**

Platform Invocation Services .NET (или PInvoke) позволяют управляемому коду .NET обращаться к функциям и структурам в существующих неуправляемых библиотеках DLL, написанных на C.

- При помощи PInvoke программист .NET избавляется от массы проблем, связанных с реализацией вызова функций и экспорта возвращаемых ими данных вручную.
- PInvoke передает параметры при вызове функций, транслируя типы данных .NET в их аналоги в традиционном двоичном коде.

Платформа .NET поддерживает следующие виды совместимости с унаследованным кодом:

- вызовы из типов .NET напрямую к модулям DLL, созданным на С (то есть обращения к Win32 API или пользовательским модулям DLL);
- вызовы из типов .NET к типам COM;
- вызовы из типов COM к типам .NET;
- вызовы из типов .NET к службам COM.

Название	Описание
CharSet	Определяет, в каком формате (Unicode/ANSI) будут передаваться вызываемой функции строковые данные
EntryPoint	Определяет имя вызываемой функции
ExactSpelling	Если это поле будет иметь значение true, то никакие попытки служб Pinvoke «догадаться», какое именно имя вызываемой функции имеется в виду, допускаться не будут: будет требоваться точное совпадение имени вызываемой функции с переданным именем
PreserveSig	Если это поле будет иметь значение true, сигнатура метода традиционной DLL не будет преобразована в сигнатуру метода .NET

Пример использования DllImport

```
// Атрибут указывает, что при вызове метода MessageBox()  
// из библиотеки user32.dll  
// вызывается функция MessageBoxA() (как наиболее подходящая)  
[DllImport("user32")]  
public static extern int MessageBox(int hWnd, String pText,  
                                     String pCaption, int uType);  
  
[DllImport("kernel32.dll", CharSet = CharSet.Unicode)]  
public static extern bool CopyFile(string lpExistingFileName,  
                                   string lpNewFileName, bool bFailIfExists);  
  
[DllImport("kernel32.dll")]  
public static extern void Sleep(uint dwMilliseconds);  
  
// Требования:  
// 1. static и extern - обязательные спецификаторы  
// 2. использование типов C#  
// 3. обязательное использование атрибута DllImport  
//    с указанием имени библиотеки
```

Пример использования DllImport

```
public static void Main()
{
    String pText = "Hello World!";
    String pCaption = "Унаследованный код!";
    MessageBox(0, pText, pCaption, 0);

    Console.WriteLine("Копирование");
    CopyFile(@"12.zip", @"33.zip", true);
    Sleep(1000);
    Console.WriteLine("Копирование завершено!");
    Console.Read();
}
```

http://www.pinvoke.net/default.aspx

The screenshot shows a web browser window with the URL `http://www.pinvoke.net/default.aspx/kernel32.CopyFile`. The page layout includes a sidebar on the left with a list of API functions, a main content area with C# sample code, and a right sidebar with an advertisement.

Left Sidebar (API Functions):

- fwpuclnt
- gdi32
- gdiplus
- getuname
- glu32
- glut32
- gsapi
- hhctrl
- hid
- hlink
- httpapi
- icmp
- imm32
- iphlpapi
- iprop
- irprops
- kernel32
- 777
- ActivateActCtx
- ActiveActCtx
- AddAtom
- AddConsoleAlias
- AddLocalAlternateComputerName
- AllocateUserPhysicalPages
- AllocConsole
- APIGetVersionEx
- AreFileApisANSI
- AssignProcessToJobObject
- AttachConsole
- AttachConsole
- BackupRead
- BackupSeek
- BackupWrite
- Beep
- BeginUpdateResource
- BindIoCompletionCallback
- BuildCommDCB
- BuildCommDCBAndTimeouts
- CallNamedPipe
- Canceled
- CancelWaitableTimer
- ChangeTimerQueueTimer
- CheckRemoteDebuggerPresent
- ClearCommBreak
- ClearCommError
- CloseHandle
- CommConfigDialog
- CompareFileTime
- CompareString
- CompareStringEx
- ConnectNamedPipe
- ConsoleFunctions
- ContinueDebugEvent
- ConvertDefaultLocale
- ConvertThreadToFiber
- COORD
- CopyFile
- CopyFileEx

Main Content Area:

Tips & Tricks:

Please add some!

Sample Code - C# - sivakumar.keerthi

```
using System;

using System.Drawing;

using System.Collections;

using System.ComponentModel;

using System.Windows.Forms;

using System.Data;

using System.Runtime.InteropServices;

using System.Diagnostics;

namespace WindowsApplication1

{

    /// <summary>
    /// Summary description for Form1.
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        [DllImport("kernel32.dll")]
        static extern bool CopyFile(string lpExistingFileName, string lpNewFileName, bool
bFailIfExists);

        private System.ComponentModel.IContainer components = null;

        public Form1()
        {
            //
            // Required for windows Form Designer support
            //
            InitializeComponent();
            //
            // TODO: Add any constructor code after InitializeComponent call
            //
        }

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        protected override void Dispose( bool disposing )
        {
            if( disposing )
            {
                if (components != null)
                {
                    components.Dispose();
                }
            }
        }
    }
}
```

Right Sidebar (Advertisement):

MEMORY PROFILER

Using Native Resources?

Easily understand .NET's unmanaged memory usage

[Find out more](#)

Пример использования DllImport

```
// Атрибут указывает, что при вызове метода MessageBoxA()  
// будет вызываться одноименная функция MessageBoxA()  
// из библиотеки user32.dll  
// ExactSpelling - устанавливает точное совпадение имени  
// вызываемой функции с переданным именем  
// ПОЭТОМУ необходимо установить имя MessageBoxA()  
[DllImport("user32", ExactSpelling=true)]  
public static extern int MessageBoxA(int hWnd, String pText,  
                                     String pCaption, int uType);  
  
// версия ANSI - MessageBoxA()  
// версия Unicode - MessageBoxW()  
  
public static void Main()  
{  
    String pText = "Hello World!";  
    String pCaption = "Унаследованный код!";  
    MessageBoxA(0, pText, pCaption, 0);  
    Console.Read();  
}
```

Пример использования DllImport

```
// Атрибуту можно указать, что при вызове метода MyMessageBox()  
// будет вызываться функция MessageBoxA() из библиотеки user32.dll  
  
[DllImport("user32", EntryPoint = "MessageBoxA")]  
public static extern int MyMessageBox(int hWnd, String pText,  
                                     String pCaption, int uType);  
  
public static void Main()  
{  
    String pText = "Hello World!";  
    String pCaption = "Унаследованный код!";  
    MyMessageBox(0, pText, pCaption, 0);  
    Console.Read();  
}
```

Пример использования DllImport

```
// Для указания кодировки для символьных данных при вызове
// внешней функции из кода традиционного модуля DLL
[DllImport("user32", CharSet = CharSet.Ansi)]
public static extern int MessageBox(int hWnd, String pText,
                                     String pCaption, int uType);

public static void Main()
{
    String pText = "Hello World!";
    String pCaption = "Унаследованный код!";
    MessageBox(0, pText, pCaption, 0);
    Console.Read();
}
```