.Net Interoperabilität

Objektorientiertes Programmieren mit .NET und C#

Thomas Hörmann

Institut für Informatik
Software & Systems Engineering

Agenda

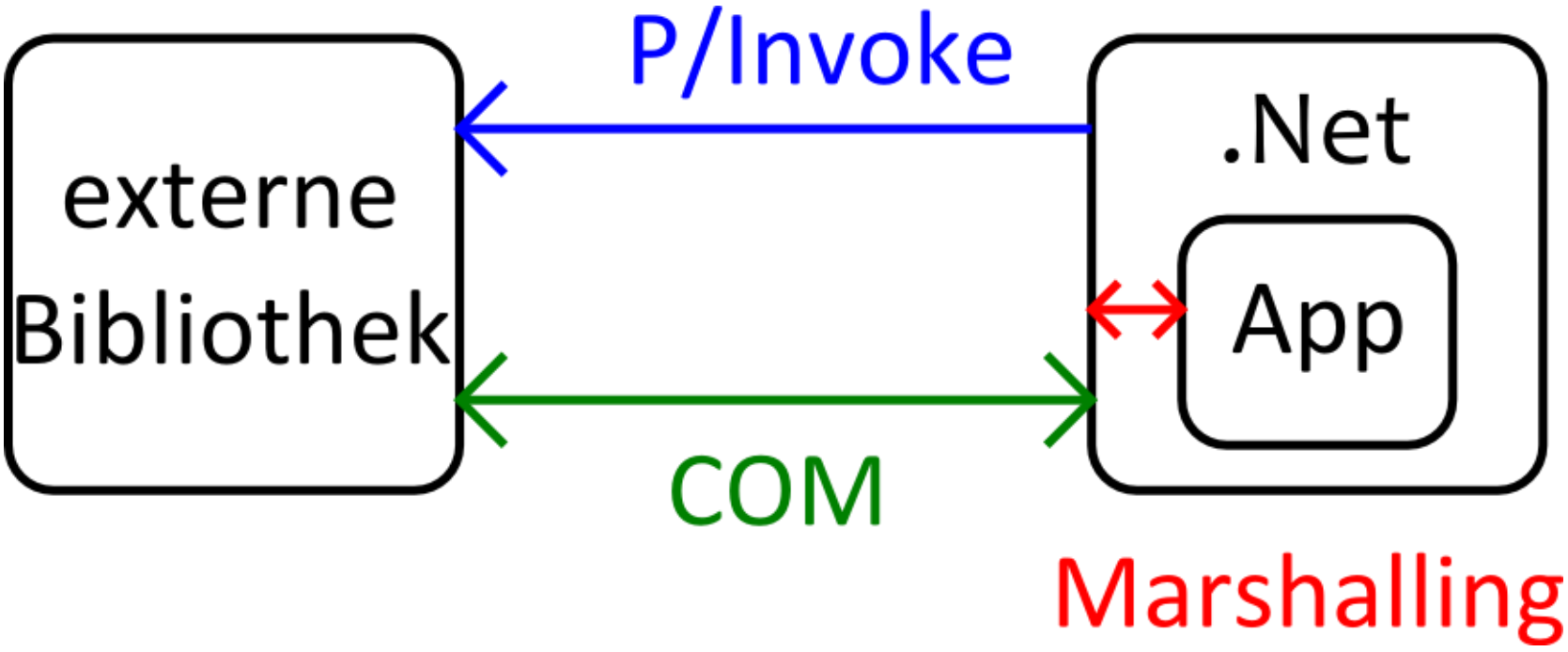
- Einleitung
- P/Invoke
- Marshalling
- Component Object Model (COM)
- Fazit

Wieso Interop?

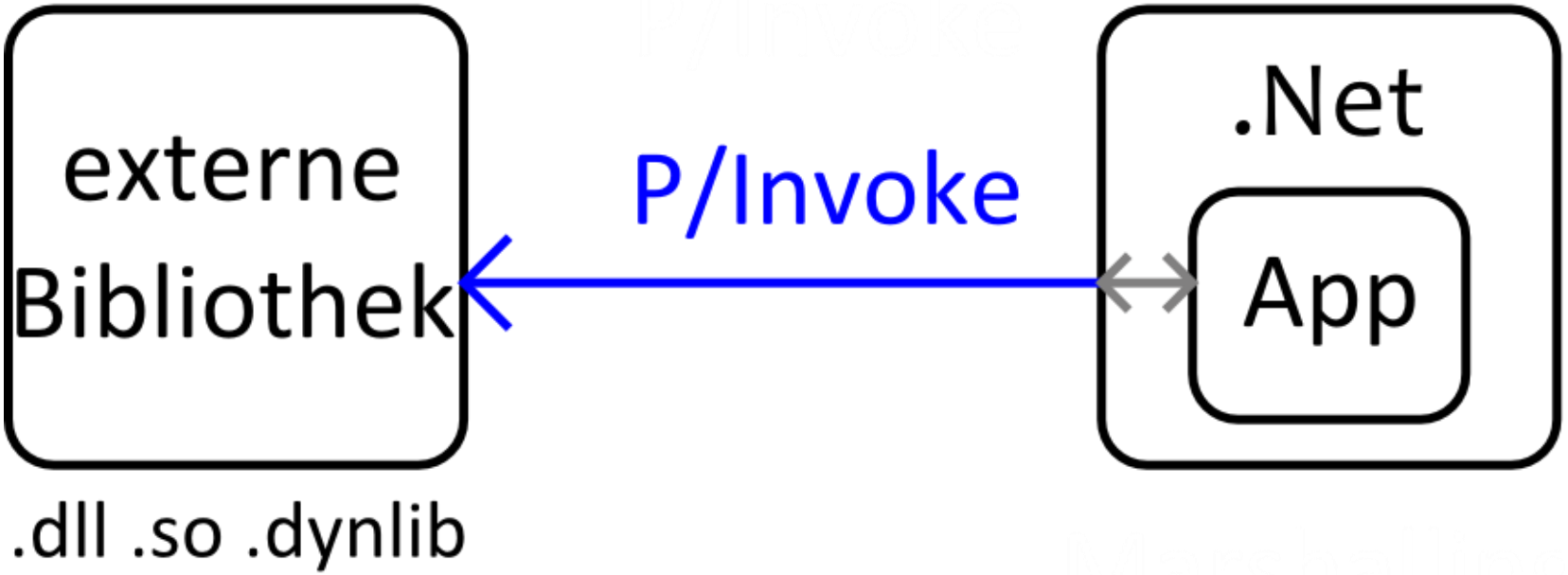
- Zugriff auf Betriebssystem Funktionen
- Bibliotheken Dritter verwenden
- Legacy Code verwenden

- Beispiele:
  - Windows 7 Glass
  - Windows Taskleiste
  - GTK
  - Hochperformanter nativer Code

Überblick



P/Invoke



DllImport

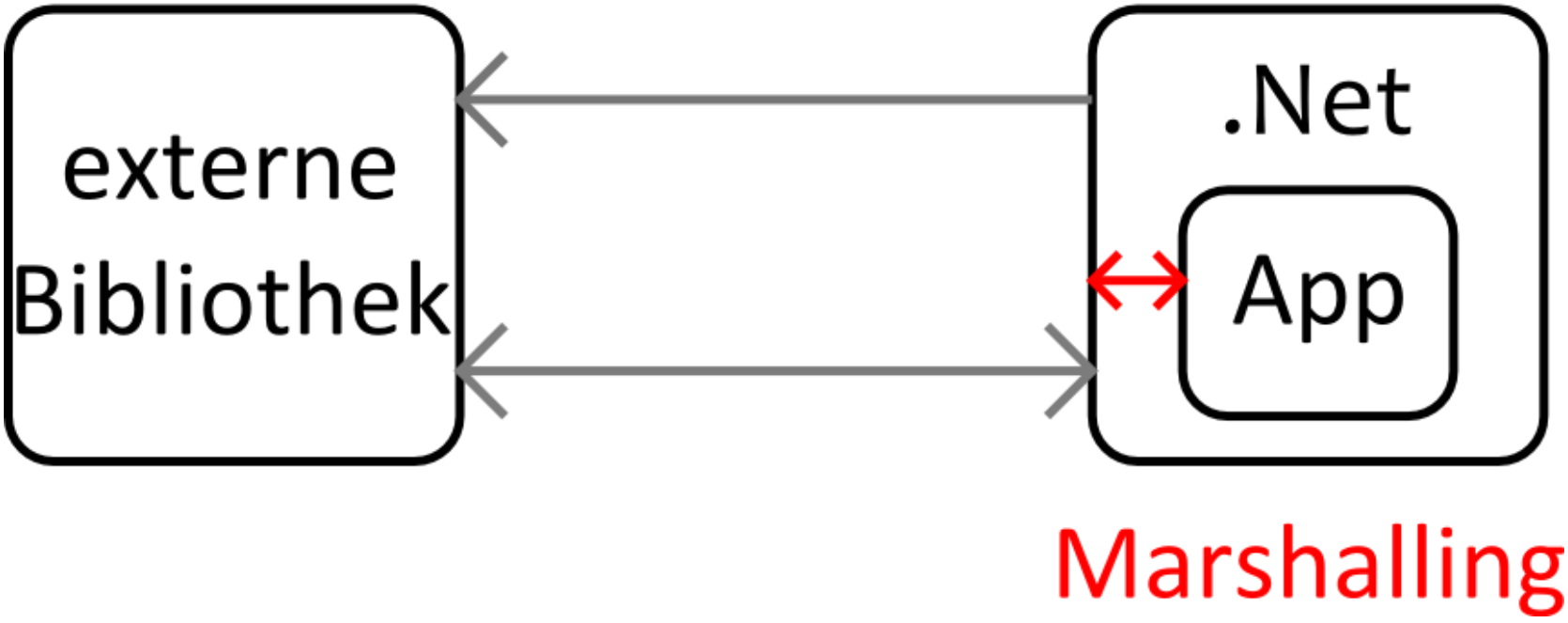
```
[DllImport("user32")]
public extern static int MessageBox(
        int hwnd, String m, String c, int flags);


[DllImport("user32.dll",
        EntryPoint="MessageBoxW",
        CallingConvention=CallingConvention.StdCall,
        ExactSpelling=true)]
public extern static uint MessageBox2(
        IntPtr hwnd, String m, String c, uint flags);
```

Fehlerquellen

- `DllNotFoundException`
- `EntryPointNotFoundException`
- `PInvokeStackInvallanceException`
- `AccessViolationException`
- `DivisionByZeroException`

Marshalling



Werttypen

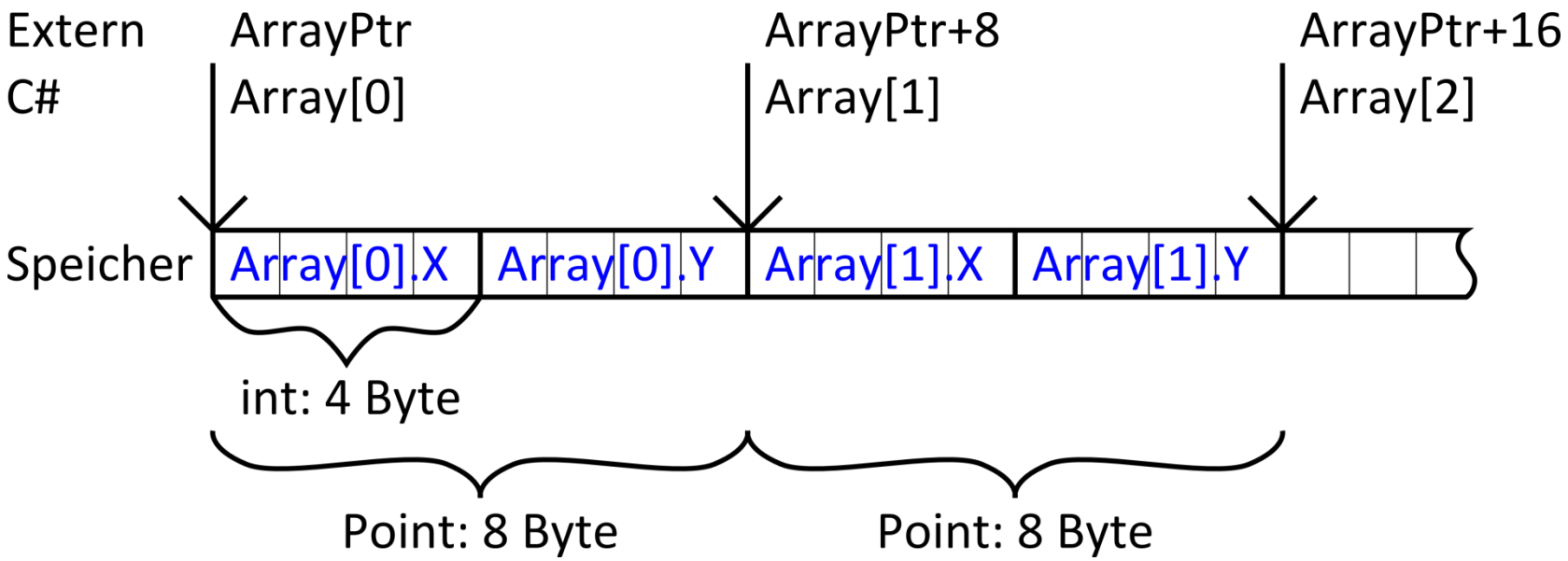
Typ	Bereich	Größe
sbyte	-128 bis 127	Ganze 8-Bit-Zahl
byte	0 bis 255	Ganze 8-Bit-Zahl
char	U+0000 bis U+ffff	16-Bit-Unicode-Zeichen
short	-32.768 bis 32.767	Ganze 16-Bit-Zahl
ushort	0 bis 65.535	Ganze 16-Bit-Zahl
int	-2.147.483.648 bis 2.147.483.647	Ganze 32-Bit-Zahl
uint	0 bis 4.294.967.295	Vorzeichenlose 32-Bit-Ganzzahl
long	-9 * 10^18 bis 9 * 10^18	64-Bit-Ganzzahl
ulong	0 bis 18 * 10^18	Vorzeichenlose 64-Bit-Ganzzahl

MarshalAs

```
[DllImport("someDll.dll")]
public static extern void DoIt(
          [MarshalAs(UnmanagedType.Bool)] int i1,
          [MarshalAs(UnmanagedType.U4)] int i2,
          [MarshalAs(UnmanagedType.LPStr)] String s1);
```

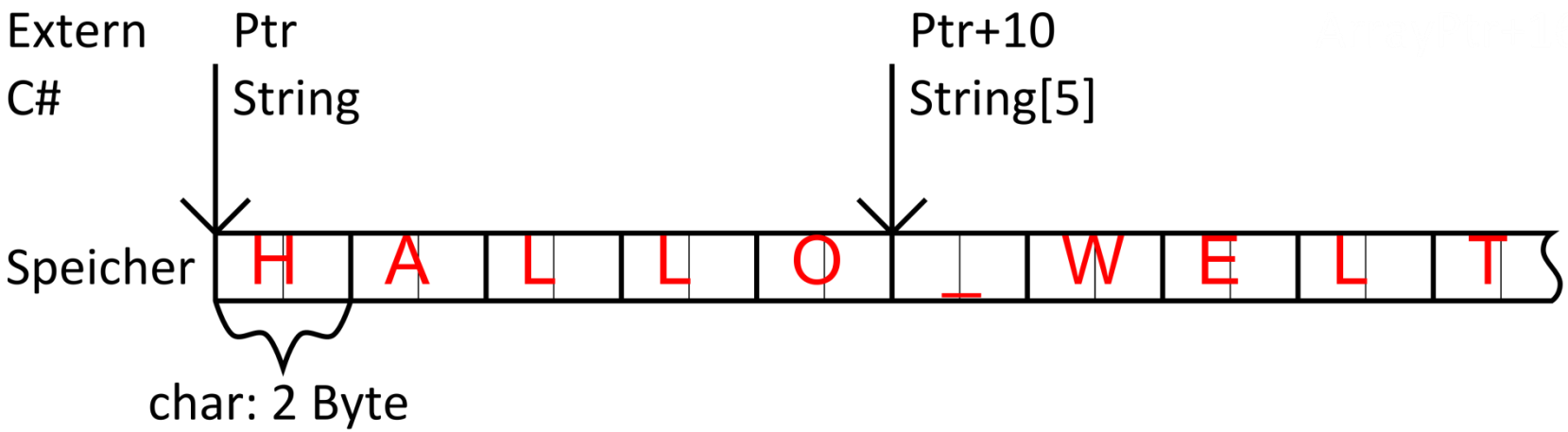
Arrays

```
[DllImport("someDll.dll")]
public static extern void doArray(POINT[] myArray);
```



Strings

```
[DllImport("someDll.dll")]
public static extern void doString(String myString);
```



Strings (2)

Probleme mit Strings:

- Strings sind nicht veränderbare Objekte
- Zeichenkodierung

Lösung:

- `StringBuilder`
- `Char Array`
- `MarshalAs`
- `CharSet`

```
[DllImport("user32.dll", CharSet = CharSet.Ansi)]
public extern static uint MessageBoxA(
        int hwnd, String m, String c, uint flags);
```

Strukturen

```
[StructLayout(LayoutKind.Sequential)]
public struct POINT
{
      public int X;
      public int Y;
}

[DllImport("someDll.dll")]
public static extern void doStruct(POINT p);
```

Strukturen sind Werttypen

X und Y kommen auf den Stack

Strukturen

```
[StructLayout(LayoutKind.Sequential)]
public struct POINT
{
      public int X;
      public int Y;
}

[DllImport("someDll.dll")]
public static extern void doStruct(POINT p);

[DllImport("someDll.dll")]
public static extern void doStruct(int X, int Y);
```

Klassen

```
[StructLayout(LayoutKind.Sequential)]
public class Point
{
      public int X;
      public int Y;
}

[DllImport("someDll.dll")]
public static extern void doStructRef(Point p);

[DllImport("someDll.dll")]
public static extern void doStructRef(ref POINT p);
```

MarshalByRef

```
[DllImport("dwmapi.dll", PreserveSig = false)]
public static extern void DwmExtendFrameIntoClientArea
          (IntPtr hwnd, ref MARGINS margins);
```

- Fenster
- Thread Handling (Timer, Mutex)
- IO Streams
- Datenbank Verbindungen

Funktionspointer

```
[StructLayout(LayoutKind.Sequential)]
public class Point
{
	public int X;
	public int Y;
	public void add(int X, int Y)
	{
		this.X += X;
		this.Y += Y;
	}
}
```

Funktionspointer (2)

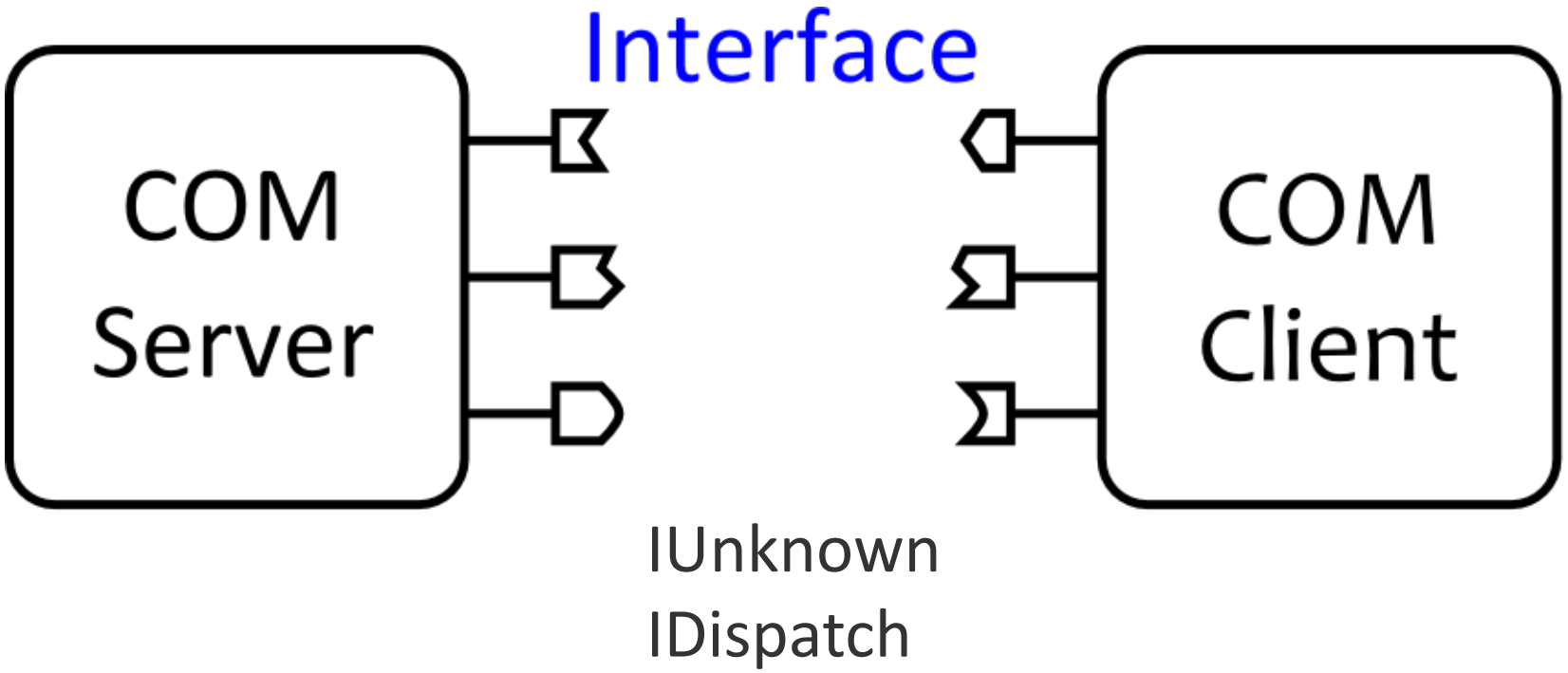
```
public delegate void addPoint(int X, int Y);

[DllImport("someDll.dll")]
public static extern void sendPointer(addPoint myDelegate);

static void Main()
{
      Point myPoint = new Point();
      addPoint myDelegate = myPoint.add;
      sendPointer(myDelegate);
}
```

Component Object Model

Component Object Model



Beispiel

Fazit

+ Einfacher Zugriff auf externe Bibliotheken
+ Einheitliche Schnittstelle
+ Volle Kontrolle über verwalteten Heap
+ Verwaltung von COM Objekten

- Keine Optimierung zur Laufzeit möglich
- Schlechte Laufzeit bei vielen Zugriffen
- Benötigt mehr Speicher
- Eingeschränkter Zugriff von außen

Vielen Dank für Ihre Aufmerksamkeit