

# Gefahren eines ReSharper „Cleanup Code“-Refactorings im Zusammenhang mit P/Invoke und übergebenen Strukturen

## ReSharper und Refactoring

Die Erweiterung „ReSharper“<sup>1</sup> bietet viele nützliche Zusatzfunktionen für Visual Studio und sollte bei der .NET-Programmierung von jedem Programmierer verwendet werden. Ein Teil der Funktionen sind gegenüber Visual Studio erweiterte Möglichkeiten zum Refactoring.<sup>2</sup> Eines der Refactorings ist „Cleanup Code“ welches unter anderem den Code nach bestimmten Styleguides automatisch formatiert.

## Problemkontext

Als ArcObjects Programmierer ist man gelegentlich gezwungen, Unmanaged Code aufzurufen, also z. B. Funktionen des Windows-APIs. So schön das automatische „Cleanup Code Refactoring“ von ReSharper auch ist – im Zusammenhang mit Strukturdefinitionen, die als Übergabeparameter für Unmanaged-Code-Aufrufe benötigt werden, lauert eine große Gefahr. Standardmäßig ist im Profil „Full Cleanup“ die Option „Reorder type members“ auf „Yes“ gestellt.

Und so wird aus der originalen Definition:

```
public struct TreeViewItem
{
    public int mask;
    public IntPtr hItem;
    public int state;
    public int stateMask;
}
```

nach dem „Cleanup Code Refactoring“ eine mit alphabetisch geordneten Type-Membem:

```
public struct TreeViewItem
{
    public IntPtr hItem;
    public int mask;
    public int state;
    public int stateMask;
}
```

Im eigenen Programmscope ist die Verwendung meistens unproblematisch – allerdings wird dies eine möglicherweise schwer zu findende Problemursache, wenn eine Instanz dieser Struktur als Parameter in einem Unmanaged-Code-Aufruf (z. B. WinAPI-Aufruf) verwendet wird. Konkret hatten wir dieses Problem mit folgendem Codefragment:

```
tvi = new TreeViewItem();
...
IntPtr lparam = Marshal.AllocHGlobal(Marshal.SizeOf(tvi));
Marshal.StructureToPtr(tvi, lparam, false);
SendMessage(Handle, (uint)TVM_SETITEM, IntPtr.Zero, lparam);
```

Seitens des Windows-API ist es nämlich festgelegt, an welcher Position der Struktur welche Informationen stehen müssen. Und der Tausch der ersten beiden Felder führt fast zwangsläufig zu Fehlverhalten bzw. Fehlern.

## Lösungsansätze

Um die Position der Felder in der Struktur fest vorzugeben – unabhängig von der Reihenfolge im Quellcode – gibt es das `[StructLayout]-Attribut`.<sup>3</sup>

Mit der Konstruktor-Variante `[StructLayout(LayoutKind.Explicit)]`<sup>4</sup> lässt sich jedes Type-Member über das `[FieldOffset]-Attribut` exakt ausrichten. So kann sichergestellt werden, dass beim (automatischen) Refactoring durch das Umordnen von Type-Membem nicht versehentlich die Semantik einer Struktur geändert wird.

ReSharper modifiziert die Reihenfolge von Type-Membem in einer Struktur prinzipiell nicht, wenn diese mit dem `[StructLayout]-Attribut` ausgezeichnet ist.<sup>5</sup> Beim ausschließlichen Einsatz von smarten Refactoring-Werkzeugen wie ReSharper kann somit die Auszeichnungsvariante `[StructLayout(LayoutKind.Sequential)]` genutzt werden, welche quasi die Standardeinstellung ist und die Positionierung der Felder so beibehält, wie sie im Quellcode als Type-Membem in der Struktur angegeben sind. Die Angabe von `[FieldOffset]-Attributen` kann dann entfallen.

Nichtsdestotrotz bleibt dann ein Restrisiko, wenn zukünftig mal andere Refactoring-Werkzeuge zum Einsatz kommen, die dieses Attribut eben nicht berücksichtigen. Einen derartigen Fehler später zu finden kann eine unglaubliche Sisyphusarbeit sein, wie wir selber schon schmerzlich erfahren mussten. Deshalb sollte der Clean Code Developer<sup>6</sup> lieber gleich Nägel mit Köpfen machen und die Ausrichtung in Eigeninitiative mittels `[StructLayout(LayoutKind.Explicit)]` spezifizieren. ++

Marko Apfel  
ESRI Deutschland GmbH  
Kranzberg  
m.apfel@esri.de  
<http://geekswithblogs.net/mapfel/>

<sup>1</sup> JetBrains ReSharper:  
<http://www.jetbrains.com/resharper/>

<sup>2</sup> Refactoring:  
<http://de.wikipedia.org/wiki/Refactoring>

<sup>3</sup> Mastering structs in C#:  
<http://stackoverflow.com/questions/1182782/c-structlayout-explicit-question>

<sup>4</sup> MSDN – LayoutKind Enumeration:  
<http://msdn.microsoft.com/en-us/library/system.runtime.interopservices.layoutkind.aspx>

<sup>5</sup> ReSharper – Disable Naming Style checks for types with StructLayout:  
<http://youtrack.jetbrains.net/issue/RSRP-110430>

<sup>6</sup> Clean Code Developer (CCD):  
<http://clean-code-developer.de/>