

System.Runtime.InteropServices.StructLayoutAttribute Class

```
[ILAsm]
.class public sealed StructLayoutAttribute extends System.Attribute

[C#]
public sealed class StructLayoutAttribute: Attribute
```

Assembly Info:

- *Name:* mscorlib
- *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- *Version:* 2.0.x.x
- *Attributes:*
 - CLSCompliantAttribute(true)

Type Attributes:

- AttributeUsageAttribute(AttributeTargets.Class | AttributeTargets.Struct, AllowMultiple=false, Inherited=false)

Summary

The `System.Runtime.InteropServices.StructLayoutAttribute` allows the user to control the physical layout of the data members of a class or structure.

Inherits From: System.Attribute

Library: RuntimeInfrastructure

Thread Safety: All public static members of this type are safe for multithreaded operations. No instance members are guaranteed to be thread safe.

Description

The target objects for this attribute are classes and structures. By default, the physical layout of the data members of a target object is automatically arranged. When managed objects are passed as arguments to unmanaged code, the system creates their unmanaged representations. These unmanaged representations can be controlled with the `System.Runtime.InteropServices.StructLayoutAttribute`. Such control is necessary if the unmanaged code expects a specific layout, packing size, or character set.

[*Note:* See the `System.Runtime.InteropServices.LayoutKind` enumeration for a description of the possible layout schemes, and the `System.Runtime.InteropServices.FieldOffsetAttribute` for further information on the layout of exported objects.]

1
2
3
4 Compilers are required to not preserve this type in metadata as a custom attribute.
5 Instead, compilers are required to emit it directly in the file format, as described in
6 Partition II of the CLI Specification. Metadata consumers, such as the Reflection API, are
7 required to retrieve this data from the file format and return it as if it were a custom
8 attribute.

9 Example

10 The following example demonstrates the use of the
11 `System.Runtime.InteropServices.StructLayoutAttribute`, and the
12 `System.Runtime.InteropServices.FieldOffsetAttribute`.

13
14 [*Note:* The non-standard `PtInRect` function used in this example indicates whether the
15 specified point is located inside the specified rectangle. In this example, the layout
16 setting on the `Rect` structure can be set to
17 `System.Runtime.InteropServices.LayoutKind.Sequential` with no bearing on the
18 end result.]

```
19  
20  
21  
22 [C#]  
  
23 using System;  
24 using System.Runtime.InteropServices;  
25  
26 [StructLayout(LayoutKind.Sequential)]  
27 public struct Point {  
28     public int x;  
29     public int y;  
30 }  
31  
32 [StructLayout(LayoutKind.Explicit)]  
33 public struct Rect {  
34     [FieldOffset(0)] public int left;  
35     [FieldOffset(4)] public int top;  
36     [FieldOffset(8)] public int right;  
37     [FieldOffset(12)] public int bottom;  
38 }  
39  
40  
41 class NativeCodeAPI {  
42     [DllImport("User32.dll")]  
43     public static extern bool PtInRect(ref Rect r, Point p);  
44 }  
45  
46 public class StructLayoutTest {  
47     public static void Main() {  
48         Rect r;  
49         Point p1, p2;  
50  
51         r.left = 0;  
52         r.right = 100;
```

```
1  r.top = 0;
2  r.bottom = 100;
3
4  p1.x = 20;
5  p1.y = 30;
6
7  p2.x = 110;
8  p2.y = 5;
9
10
11  bool isInside1 = NativeCodeAPI.PtInRect(ref r, p1);
12  bool isInside2 = NativeCodeAPI.PtInRect(ref r, p2);
13
14  if(isInside1)
15  Console.WriteLine("The first point is inside the rectangle.");
16  else
17  Console.WriteLine("The first point is outside the rectangle.");
18
19  if(isInside2)
20  Console.WriteLine("The second point is inside the rectangle.");
21  else
22  Console.WriteLine("The second point is outside the rectangle.");
23
24  }
25  }
26  The output is
27
28  The first point is inside the rectangle.
29
30
31  The second point is outside the rectangle.
32
33
```

1

2 StructLayoutAttribute(System.Runtime.InteropServices.LayoutKind) Constructor

3

```
4 [ILAsm]  
5 public rtspecialname specialname instance void .ctor(valuetype  
6 System.Runtime.InteropServices.LayoutKind layoutKind)  
  
7 [C#]  
8 public StructLayoutAttribute(LayoutKind layoutKind)
```

9 **Summary**

10 Constructs and initializes a new instance of the
11 System.Runtime.InteropServices.StructLayoutAttribute class with the specified
12 System.Runtime.InteropServices.LayoutKind value.

13 **Parameters**

Parameter	Description
<i>layoutKind</i>	A System.Runtime.InteropServices.LayoutKind value that specifies how the class or structure is arranged in memory.

14

15 **Description**

16 If *layoutKind* contains an invalid System.Runtime.InteropServices.LayoutKind value,
17 a runtime error occurs.

18

1 StructLayoutAttribute(System.Int16)

2 Constructor

```
3 [ILAsm]  
4 public rtspecialname specialname instance void .ctor(int16 layoutKind)  
5 [C#]  
6 public StructLayoutAttribute(short layoutKind)
```

7 Summary

8 Constructs and initializes a new instance of the
9 System.Runtime.InteropServices.StructLayoutAttribute class with the specified
10 value.

11 Parameters

Parameter	Description
<i>layoutKind</i>	A System.Int16 set to a System.Runtime.InteropServices.LayoutKind value that specifies how the class or structure is arranged in memory.

12 13 Description

14 If the *layoutKind* parameter does not represent a valid
15 System.Runtime.InteropServices.LayoutKind value, a runtime error occurs.

16

1 StructLayoutAttribute.CharSet Field

```
2 [ILAsm]  
3 .field public valuetype System.Runtime.InteropServices.CharSet CharSet  
4 [C#]  
5 public CharSet CharSet
```

6 Summary

7 A System.Runtime.InteropServices.CharSet value that indicates the character set in
8 which strings of an object are marshaled.

9 Description

10 [*Note:* See the System.Runtime.InteropServices.CharSet enumeration for a
11 description of different character sets.]
12
13

14 The default value of this field is System.Runtime.InteropServices.CharSet.Ansi.
15
16

1 StructLayoutAttribute.Pack Field

```
2 [ILAsm]  
3 .field public int32 Pack  
4 [C#]  
5 public int Pack
```

6 Summary

7 A `System.Int32` that indicates the packing alignment used with the
8 `System.Runtime.InteropServices.LayoutKind.Sequential` layout.

9 Description

10 The `System.Runtime.InteropServices.StructLayoutAttribute.Pack` field determines
11 memory alignment of data fields of a target object.

12
13 Data fields of a target object exported to unmanaged memory are aligned on a byte
14 boundary that is a multiple of
15 `System.Runtime.InteropServices.StructLayoutAttribute.Pack` bytes, or at some
16 natural alignment for that field type, whichever is smaller.

17
18 The value of `System.Runtime.InteropServices.StructLayoutAttribute.Pack` is
19 required to be 0, 1, 2, 4, 8, 16, 32, 64, or 128. A value of zero indicates that the
20 packing alignment is set to the default for the current platform. The default value is
21 implementation-defined.

22

1 StructLayoutAttribute.Size Field

```
2 [ILAsm]  
3 .field public int32 Size  
4 [C#]  
5 public int Size
```

6 Summary

7 A `System.Int32` that indicates the size of the memory block to be allocated for an
8 instance of the type qualified by the current
9 `System.Runtime.InteropServices.StructLayoutAttribute`.

10 Description

11 `System.Runtime.InteropServices.StructLayoutAttribute.Size` is required to be
12 zero, or greater than or equal to the calculated size of the target object, based on the
13 `System.Runtime.InteropServices.StructLayoutAttribute.Pack` field indicating the
14 packing alignment. A
15 `System.Runtime.InteropServices.StructLayoutAttribute.Size` of zero indicates
16 that the size is calculated from the field types, their specified offsets, the packing size
17 (default or specified) and natural alignment on the target, runtime platform.

18
19 [*Note:* For additional information on the
20 `System.Runtime.InteropServices.StructLayoutAttribute.Size` field, see Partition II
21 of the CLI Specification.]
22
23

24

1 StructLayoutAttribute.Value Property

```
2 [ILAsm]  
3 .property valuetype System.Runtime.InteropServices.LayoutKind Value {  
4 public hidebysig specialname instance valuetype  
5 System.Runtime.InteropServices.LayoutKind get_Value() }  
  
6 [C#]  
7 public LayoutKind Value { get; }
```

8 Summary

9 Gets the `System.Runtime.InteropServices.LayoutKind` value that specifies how the
10 target object is arranged.

11 Property Value

12 A `System.Runtime.InteropServices.LayoutKind` value that specifies how the target
13 object is arranged.

14 Description

15 This property is read-only.

16