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.]

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

Example

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

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

[C#]

```
using System;
using System.Runtime.InteropServices;

[StructLayout(LayoutKind.Sequential)]
public struct Point {
 public int x;
 public int y;
}

[StructLayout(LayoutKind.Explicit)]
public struct Rect {
 [FieldOffset(0)] public int left;
 [FieldOffset(4)] public int top;
 [FieldOffset(8)] public int right;
 [FieldOffset(12)] public int bottom;
}


class NativeCodeAPI {
 [DllImport("User32.dll")]
 public static extern bool PtInRect(ref Rect r, Point p);
}

public class StructLayoutTest {
 public static void Main() {
 Rect r;
 Point p1, p2;
 r.left = 0;
 r.right = 100;
 r.top = 0;
 r.bottom = 100;

 p1.x = 20;
 p1.y = 30;

 p2.x = 110;
 p2.y = 5;


 bool isInside1 = NativeCodeAPI.PtInRect(ref r, p1);
 bool isInside2 = NativeCodeAPI.PtInRect(ref r, p2);

 if(isInside1)
 Console.WriteLine("The first point is inside the rectangle.");
 else
 Console.WriteLine("The first point is outside the rectangle.");

 if(isInside2)
 Console.WriteLine("The second point is inside the rectangle.");
 else
 Console.WriteLine("The second point is outside the rectangle.");

 }
}
```

The output is

```
The first point is inside the rectangle.


The second point is outside the rectangle.
```

StructLayoutAttribute(System.Runtime.InteropServices.LayoutKind) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(valuetype
System.Runtime.InteropServices.LayoutKind layoutKind)

[C#]
public StructLayoutAttribute(LayoutKind layoutKind)
```

Summary

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

Parameters

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

Description

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

StructLayoutAttribute(System.Int16) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(int16
layoutKind)

[C#]
public StructLayoutAttribute(short layoutKind)
```

Summary

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

Parameters

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

Description

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

StructLayoutAttribute.CharSet Field

[ILAsm]
```
.field public valuetype System.Runtime.InteropServices.CharSet
CharSet
```

[C#]
```
public CharSet CharSet
```

Summary

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

Description

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

The default value of this field is `System.Runtime.InteropServices.CharSet.Ansi`.

StructLayoutAttribute.Pack Field

[ILAsm]
```
.field public int32 Pack
```

[C#]
```
public int Pack
```

Summary

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

Description

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

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

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

StructLayoutAttribute.Size Field

[ILAsm]
```
.field public int32 Size
```

[C#]
```
public int Size
```

Summary

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

Description

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

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

StructLayoutAttribute.Value Property

[ILAsm]
```
.property valuetype System.Runtime.InteropServices.LayoutKind Value
{ public hidebysig specialname instance valuetype
System.Runtime.InteropServices.LayoutKind get_Value() }
```

[C#]
```
public LayoutKind Value { get; }
```

Summary

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

Property Value

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

Description

This property is read-only.