System.Reflection.MethodBase Class

```
[ILASM]
.class public abstract serializable MethodBase extends
System.Reflection.MemberInfo

[C#]
public abstract class MethodBase: MemberInfo
```

Assembly Info:

- *Name:* mscorlib
- *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- *Version:* 1.0.x.x
- *Attributes:*
  - CLSCompliantAttribute(true)

Summary

Provides access to method and constructor metadata.

Inherits From: System.Reflection.MemberInfo

Library: Reflection

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

Description

[*Note:* **MethodBase** is used to represent method types.

The Base Class Library includes the following derived types:

- **System.Reflection.MethodInfo**
- **System.Reflection.ConstructorInfo**

]

MethodBase() Constructor

[ILASM]
```
family rtspecialname specialname instance void .ctor()
```

[C#]
```
protected MethodBase()
```

Summary

Constructs a new instance of the **System.Reflection.MethodBase** class.

The following member must be implemented if the RuntimeInfrastructure library is present in the implementation.

MethodBase.GetMethodFromHandle(System.RuntimeMethodHandle) Method

```
[ILASM]
.method public hidebysig static class
System.Reflection.MethodBase GetMethodFromHandle(valuetype
System.RuntimeMethodHandle handle)

[C#]
public static MethodBase
GetMethodFromHandle(RuntimeMethodHandle handle)
```

Summary

Gets method information by using the method's internal metadata representation (handle).

Parameters

Parameter	Description
handle	The method's **System.RuntimeMethodHandle** handle.

Return Value

A **System.Reflection.MethodBase** object containing information about the method.

Description

The handles are valid only in the application domain in which they were obtained.

MethodBase.GetParameters() Method

```
[ILASM]
.method public hidebysig virtual abstract class
System.Reflection.ParameterInfo[] GetParameters()

[C#]
public abstract ParameterInfo[] GetParameters()
```

Summary

Returns the parameters of the method or constructor reflected by the current instance.

Return Value

An array of **System.Reflection.ParameterInfo** objects that contain information that matches the signature of the method or constructor reflected by the current instance.

Behaviors

As described above.

MethodBase.Invoke(System.Object, System.Reflection.BindingFlags, System.Reflection.Binder, System.Object[], System.Globalization.CultureInfo) Method

```
[ILASM]
.method public hidebysig virtual abstract object
Invoke(object obj, valuetype System.Reflection.BindingFlags
invokeAttr, class System.Reflection.Binder binder, class
System.Object[] parameters, class
System.Globalization.CultureInfo culture)

[C#]
public abstract object Invoke(object obj, BindingFlags
invokeAttr, Binder binder, object[] parameters, CultureInfo
culture)
```

Summary

Invokes the method or constructor reflected by the current instance as determined by the specified arguments.

Parameters

Parameter	Description
obj	An instance of the type that contains the method reflected by the current instance. If the method is static, *obj* is ignored. For non-static methods, *obj* is an instance of a class that inherits or declares the method.
invokeAttr	A **System.Reflection.BindingFlags** value that controls the binding process.
binder	An object that enables the binding, coercion of argument types, invocation of members, and retrieval of **MemberInfo** objects via reflection. If *binder* is **null**, the default binder is used.
parameters	An array of objects that match the number, order and type of the parameters for the constructor or method reflected by the current instance. If the member reflected by the current instance takes no parameters, specify either an array with zero elements or **null**. [*Note:* Any object in this array that is not explicitly initialized with a value will contain the default value for that object type. For reference-type elements, this value is **null**. For value-type elements, this value is 0, 0.0, or **false**, depending on the specific element type. If the method or constructor reflected by the current instance is **static**, this parameter is ignored.]

culture	The only defined value for this parameter is **null**.

Return Value

A **System.Object** that contains the return value of the invoked method, or a re-initialized object if a constructor was invoked.

Description

Optional parameters can not be omitted in calls to **System.Reflection.MethodBase.Invoke**.

Exceptions

Exception	Condition
System.ArgumentException	The types of the elements of *parameters* do not match the types of the parameters accepted by the constructor or method reflected by the current instance, under the constraints of *binder*.
System.Reflection.TargetException	The constructor or method reflected by the current instance is non-static, and *obj* is **null** or is of a type that does not implement the member reflected by the current instance.
System.Reflection. TargetInvocationException	The method reflected by the current instance threw an exception.
System.Reflection. TargetParameterCountException	*parameters*.Length does not equal the number of parameters required by the contract of the constructor or method reflected by the current instance.

Permissions

Permission	Description
System.Security.Permissions. ReflectionPermission	Requires permission to invoke non-public members of loaded assemblies. See **System.Security.Permissions. ReflectionPermissionFlag.MemberAccess**.

MethodBase.Invoke(System.Object, System.Object[]) Method

```
[ILASM]
.method public hidebysig instance object Invoke(object obj,
class System.Object[] parameters)

[C#]
public object Invoke(object obj, object[] parameters)
```

Summary

Invokes the method or constructor reflected by the current instance on the specified object and using the specified arguments.

Parameters

Parameter	Description
obj	An instance of a type that contains the constructor or method reflected by the current instance. If the member is static, *obj* is ignored. For non-static methods, *obj* is an instance of a class that inherits or declares the method.
parameters	An array objects that match the number, order and type of the parameters for the constructor or method reflected by the current instance. If the member reflected by the current instance takes no parameters, specify either an array with zero elements or **null**. [*Note:* Any object in this array that is not explicitly initialized with a value will contain the default value for that object type. For reference-type elements, this value is **null**. For value-type elements, this value is 0, 0.0, or **false**, depending on the specific element type. If the method or constructor reflected by the current instance is **static**, this parameter is ignored.]

Return Value

A **System.Object** that contains the return value of the invoked method, or a re-initialized object if a constructor was invoked.

Description

This version of **System.Reflection.MethodBase.Invoke** is equivalent to **System.Reflection.MethodBase.Invoke**(*obj*, (**BindingFlags**)0, **null**, *parameters*, **null**).

Optional parameters can not be omitted in calls to **System.Reflection.MethodBase.Invoke**.

Exceptions

Exception	Condition
System.ArgumentException	The types of the elements of *parameters* do not match the types of the parameters accepted by the constructor or method reflected by the current instance, under the constraints of *binder*.
System.Reflection.TargetException	The constructor or method reflected by the current instance is non-static, and *obj* is **null** or is of a type that does not implement the member reflected by the current instance.
System.Reflection.TargetInvocationException	The constructor or method reflected by the current instance threw an exception.
System.Reflection.TargetParameterCountException	*parameters*.Length does not equal the number of parameters required by the contract of the member reflected by the current instance.

Permissions

Permission	Description
System.Security.Permissions.ReflectionPermission	Requires permission to invoke non-public members of loaded assemblies. See **System.Security.Permissions.ReflectionPermissionFlag.MemberAccess**.

MethodBase.Attributes Property

[ILASM]

```
.property valuetype System.Reflection.MethodAttributes
Attributes { public hidebysig virtual abstract specialname
valuetype System.Reflection.MethodAttributes
get_Attributes() }
```

[C#]

```
public abstract MethodAttributes Attributes { get; }
```

Summary

Gets the attributes of the method reflected by the current instance.

Property Value

A **System.Reflection.MethodAttributes** value that signifies the attributes of the method reflected by the current instance.

Behaviors

This property is read-only.

This property gets a **System.Reflection.MethodAttributes** value that indicates the attributes set in the metadata of the method reflected by the current instance.

Usage

Use this property to determine the accessibility, layout, and semantics of the constructor or method reflected by the current instance. Also use this property to determine if the member reflected by the current instance is implemented in native code or has a special name.

Example

The following example demonstrates using this property to obtain the attributes of three methods.

[C#]

```
using System;
using System.Reflection;

abstract class MyBaseClass
{

   abstract public void MyPublicInstanceMethod();
```

```
1
2          }
3
4          class MyDerivedClass: MyBaseClass
5          {
6
7             public override void MyPublicInstanceMethod() {}
8             private static void MyPrivateStaticMethod() {}
9
10         }
11
12         class MethodAttributesExample
13         {
14
15            static void PrintMethodAttributes(Type t)
16            {
17
18               string str;
19               MethodInfo[] miAry = t.GetMethods(BindingFlags.Static
20         |
21                  BindingFlags.Instance | BindingFlags.Public |
22                  BindingFlags.NonPublic |
23         BindingFlags.DeclaredOnly);
24               foreach (MethodInfo mi in miAry)
25               {
26
27                  Console.WriteLine("Method {0} is: ", mi.Name);
28                  str = ((mi.Attributes & MethodAttributes.Static)
29         != 0) ?
30                     "Static": "Instance";
31                  Console.Write(str + " ");
32                  str = ((mi.Attributes & MethodAttributes.Public)
33         != 0) ?
34                     "Public": "Not-Public";
35                  Console.Write(str + " ");
36                  str = ((mi.Attributes &
37         MethodAttributes.HideBySig) != 0) ?
38                     "HideBySig": "Hide-by-name";
39                  Console.Write(str + " ");
40                  str = ((mi.Attributes & MethodAttributes.Abstract)
41         != 0) ?
42                     "Abstract": String.Empty;
43                  Console.WriteLine(str);
44
45               }
46
47            }
48
49            public static void Main()
50            {
51
52               PrintMethodAttributes(typeof(MyBaseClass));
53               PrintMethodAttributes(typeof(MyDerivedClass));
54
55            }
56
57         }
```

The output is

```
Method MyPublicInstanceMethod is:


Instance Public HideBySig Abstract


Method MyPublicInstanceMethod is:


Instance Public HideBySig


Method MyPrivateStaticMethod is:


Static Not-Public HideBySig
```