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:* 2.0.x.x
- *Attributes:*
  - CLSCompliantAttribute(true)

Summary

Provides information about methods and constructors.

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 Reflection library is present in the implementation.

MethodBase.GetGenericArguments() Method

```
[ILAsm]
.method public hidebysig virtual class System.Type[] GetGenericArguments()

[C#]
public virtual Type[] GetGenericArguments()
```

Summary

Returns an array of System.Type objects that represent the type arguments of a generic method or the type parameters of a generic method definition.

Return Value

An array of System.Type objects that represent the type arguments of a generic method or the type parameters of a generic method definition. Returns an empty array if the current method is not a generic method.

Description

The default behavior, when not overridden in a derived class, is to throw System.NotSupportedException. In other words, derived classes do not support generics by default.

The elements of the returned array are in the order in which they appear in the list of type parameters for the generic method.

- If the current method is a closed constructed method (that is, the System.Reflection.MethodBase.ContainsGenericParameters property returns false), the array returned by the System.Reflection.MethodBase.GetGenericArguments method contains the types that have been assigned to the generic type parameters of the generic method definition.

- If the current method is a generic method definition, the array contains the type parameters.

- If the current method is an open constructed method (that is, the System.Reflection.MethodBase.ContainsGenericParameters property returns true) in which specific types have been assigned to some type parameters and type parameters of enclosing generic types have been assigned to other type parameters, the array contains both types and type parameters. Use the System.Type.IsGenericParameter property to tell them apart.

For a list of the invariant conditions for terms specific to generic methods, see the `System.Reflection.MethodInfo.IsGenericMethod` property. For a list of the invariant conditions for other terms used in generic reflection, see the `System.Type.IsGenericType` property.

Exceptions

Exception	Condition
System.NotSupportedException	Default behavior when not overridden in a derived 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, 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 the default 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.
System.MemberAccessException	The caller does not have permission to execute the method or constructor.

System.InvalidOperationException	The type that declares the method is an open generic type. That is, `System.Type.ContainsGenericParameters` returns `true` for the declaring type.

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, 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 cannot 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 the default 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.
System.MemberAccessException	The caller does not have permission to execute the method or constructor.
System.InvalidOperationException	The type that declares the method is an open generic type. That is, System.Type.ContainsGenericParameters returns true for the declaring type.

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();

}

class MyDerivedClass: MyBaseClass
{

   public override void MyPublicInstanceMethod() {}
   private static void MyPrivateStaticMethod() {}

}

class MethodAttributesExample
{

   static void PrintMethodAttributes(Type t)
   {

      string str;
      MethodInfo[] miAry = t.GetMethods( BindingFlags.Static |
         BindingFlags.Instance | BindingFlags.Public |
         BindingFlags.NonPublic | BindingFlags.DeclaredOnly );
      foreach (MethodInfo mi in miAry)
      {

         Console.WriteLine("Method {0} is: ", mi.Name);
         str = ((mi.Attributes & MethodAttributes.Static) != 0) ?
            "Static": "Instance";
         Console.Write(str + " ");
         str = ((mi.Attributes & MethodAttributes.Public) != 0) ?
            "Public": "Not-Public";
         Console.Write(str + " ");
         str = ((mi.Attributes & MethodAttributes.HideBySig) != 0) ?
            "HideBySig": "Hide-by-name";
         Console.Write(str + " ");
         str = ((mi.Attributes & MethodAttributes.Abstract) != 0) ?
            "Abstract": String.Empty;
         Console.WriteLine(str);

      }

   }

   public static void Main()
   {

      PrintMethodAttributes(typeof(MyBaseClass));
      PrintMethodAttributes(typeof(MyDerivedClass));

   }

}

The output is

Method MyPublicInstanceMethod is:
```

```
1
2
3   Instance Public HideBySig Abstract
4
5
6   Method MyPublicInstanceMethod is:
7
8
9   Instance Public HideBySig
10
11
12  Method MyPrivateStaticMethod is:
13
14
15  Static Not-Public HideBySig
16
17
```

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

MethodBase.ContainsGenericParameters Property

```
[ILAsm]
.property bool ContainsGenericParameters { public hidebysig virtual
specialname bool get_ContainsGenericParameters() }

[C#]
public virtual bool ContainsGenericParameters { get; }
```

Summary

Gets a value that indicates whether a generic method contains unassigned generic type parameters.

Property Value

`true` if the current method contains unassigned generic type parameters; otherwise `false`.

Description

The default behavior, when not overridden in a derived class, is to return `false`. In other words, by default, derived classes do not support generics.

In order to invoke a generic method, there must be no generic type definitions or open constructed types in the type arguments of the method itself, or in any enclosing types. If the `System.Reflection.MethodBase.ContainsGenericParameters` property returns `true`, the method cannot be invoked.

The `System.Reflection.MethodBase.ContainsGenericParameters` property searches recursively for type parameters. For example, it returns `true` for any method in an open type A<T>, even though the method itself is not generic. Contrast this with the behavior of the `System.Reflection.MethodBase.IsGenericMethod` property, which returns `false` for such a method.

For a list of the invariant conditions for terms specific to generic methods, see the `System.Reflection.MethodInfo.IsGenericMethod` property. For a list of the invariant conditions for other terms used in generic reflection, see the `System.Type.IsGenericType` property.

Behaviors

This property is read-only.

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

MethodBase.IsGenericMethod Property

```
[ILAsm]
.property bool IsGenericMethod { public hidebysig virtual specialname bool
get_IsGenericMethod() }

[C#]
public virtual bool IsGenericMethod { get; }
```

Summary

Gets a value that indicates whether the current object is a generic method.

Property Value

`true` if the current object is a generic method; otherwise `false`.

Description

The default behavior, when not overridden in a derived class, is to return `false`. In other words, by default, derived classes do not support generics.

Use this property to determine whether the current `System.Reflection.MethodBase` object represents a generic method. Use the `System.Reflection.MethodBase.ContainsGenericParameters` property to determine whether the current `System.Reflection.MethodBase` object represents an open constructed method or a closed constructed method.

For a list of the invariant conditions for terms specific to generic methods, see the `System.Reflection.MethodInfo.IsGenericMethod` property. For a list of the invariant conditions for other terms used in generic reflection, see the `System.Type.IsGenericType` property.

Behaviors

This property is read-only.

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

MethodBase.IsGenericMethodDefinition Property

```
[ILAsm]
.property bool IsGenericMethodDefinition { public hidebysig virtual
specialname bool get_IsGenericMethodDefinition() }

[C#]
public virtual bool IsGenericMethodDefinition { get; }
```

Summary

Gets a value that indicates whether the current `System.Reflection.MethodBase` represents a definition of a generic method.

Property Value

`true` if the current `System.Reflection.MethodBase` object represents the definition of a generic method; otherwise `false`.

Description

The default behavior, when not overridden in a derived class, is to return `false`. In other words, by default, derived classes do not support generics.

If the current `System.Reflection.MethodBase` represents a generic method definition, then:

- `System.Reflection.MethodBase.IsGenericMethodDefinition` returns `true`.
- For each `System.Type` object in the array returned by the `System.Reflection.MethodBase.GetGenericArguments` method: The `System.Type.IsGenericParameter` property returns `true`; the `System.Type.DeclaringMethod` returns the current instance; and the `System.Type.GenericParameterPosition` property is the same as the position of the `System.Type` object in the array.

For a list of the invariant conditions for terms specific to generic methods, see the `System.Reflection.MethodInfo.IsGenericMethod` property. For a list of the invariant conditions for other terms used in generic reflection, see the `System.Type.IsGenericType` property.

Behaviors

This property is read-only.