

# 1 System.Reflection.FieldInfo Class

```
2 [ILAsm]  
3 .class public abstract serializable FieldInfo extends  
4 System.Reflection.MemberInfo  
  
5 [C#]  
6 public abstract class FieldInfo: MemberInfo
```

## 7 Assembly Info:

- 8 • *Name:* mscorlib
- 9 • *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00]
- 10 • *Version:* 2.0.x.x
- 11 • *Attributes:*
  - 12 ○ CLSCompliantAttribute(true)

## 13 Summary

14 Provides access to field metadata.

## 15 Inherits From: System.Reflection.MemberInfo

16

17 **Library:** Reflection

18

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

21

## 22 Permissions

Permission	Description

23

24

# 1 FieldInfo() Constructor

```
2 [ILAsm]  
3 family rtspecialname specialname instance void .ctor()  
4 [C#]  
5 protected FieldInfo()
```

## 6 Summary

7 Constructs a new instance of the System.Reflection.FieldInfo class.

8

# 1 FieldInfo.GetValue(System.Object) Method

```
2 [ILAsm]  
3 .method public hidebysig virtual abstract object GetValue(object obj)  
4 [C#]  
5 public abstract object GetValue(object obj)
```

## 6 Summary

7 Obtains the value of the field that is reflected by the current instance and contained in  
8 the specified object.

## 9 Parameters

Parameter	Description
<i>obj</i>	An object that contains the field value to be returned. If the field reflected by the current instance is static, <i>obj</i> is ignored. For non-static fields, <i>obj</i> is required to be an instance of a class that inherits or declares the field.

10

## 11 Return Value

12 A `System.Object` that contains the value of the field reflected by the current instance.

## 13 Behaviors

14 Before returning the value, the system checks to see if the user has access permission.

15

## 16 Exceptions

Exception	Condition
<b>System.NotSupportedException</b>	A field is marked literal, but the field does not have one of the accepted literal types. [ <i>Note:</i> For information regarding the accepted literal types, see Partition II of the CLI Specification.]
<b>System.FieldAccessException</b>	The field reflected by the current instance is non-public, and the caller does not have permission to access non-public members.

<b>System.ArgumentException</b>	The field reflected by the current instance is declared neither directly in <i>obj</i> nor in any class from which <i>obj</i> derives.
<b>System.Reflection.TargetException</b>	The field reflected by the current instance is non-static, and <i>obj</i> is null.

1

2 **Permissions**

Permission	Description
<b>System.Security.Permissions.ReflectionPermission</b>	Requires permission to access non-public members of a type in loaded assemblies. See <code>System.Security.Permissions.ReflectionPermissionFlag.MemberAccess</code> .

3

4

# 1 FieldInfo.SetValue(System.Object, 2 System.Object, 3 System.Reflection.BindingFlags, 4 System.Reflection.Binder, 5 System.Globalization.CultureInfo) Method

```
6 [ILAsm]  
7 .method public hidebysig virtual abstract void SetValue(object obj, object  
8 value, valuetype System.Reflection.BindingFlags invokeAttr, class  
9 System.Reflection.Binder binder, class System.Globalization.CultureInfo  
10 culture)  
11 [C#]  
12 public abstract void SetValue(object obj, object value, BindingFlags  
13 invokeAttr, Binder binder, CultureInfo culture)
```

## 14 Summary

15 Assigns the specified value to the field that is reflected by the current instance and  
16 contained in the specified object.

## 17 Parameters

Parameter	Description
<i>obj</i>	The object whose field value will be set. If the field is static, <i>obj</i> is ignored. For non-static fields, <i>obj</i> is required to be an instance of a class that inherits or declares the field.
<i>value</i>	An object that contains the value to assign to the field contained by <i>obj</i> .
<i>invokeAttr</i>	A System.Reflection.BindingFlags value that controls the binding process.
<i>binder</i>	A System.Reflection.Binder instance that enables the binding, coercion of argument types, and invocation of members through reflection. If <i>binder</i> is null, the default binder of the current implementation is used.
<i>culture</i>	The only defined value for this parameter is null.

18

## 19 Behaviors

20 Before setting the value, the system verifies that the user has access permission.

1

## 2 Exceptions

Exception	Condition
<b>System.ArgumentException</b>	The field reflected by the current instance is declared neither directly in <i>obj</i> nor in any class from which <i>obj</i> derives.  <i>value</i> is not assignment-compatible with the type of the field reflected by the current instance.
<b>System.FieldAccessException</b>	The field reflected by the current instance is non-public, and the caller does not have permission to access non-public members.
<b>System.Reflection.TargetException</b>	The field reflected by the current instance is non-static, and <i>obj</i> is null.

3

## 4 Permissions

Permission	Description
<b>System.Security.Permissions.ReflectionPermission</b>	Requires permission to access non-public members of a type in loaded assemblies. See <code>System.Security.Permissions.ReflectionPermissionFlag.MemberAccess</code> .

5

6

# 1 FieldInfo.SetValue(System.Object, 2 System.Object) Method

```
3 [ILAsm]  
4 .method public hidebysig instance void SetValue(object obj, object value)  
5 [C#]  
6 public void SetValue(object obj, object value)
```

## 7 Summary

8 Assigns the specified value to the field that is reflected by the current instance and  
9 contained in the specified object.

## 10 Parameters

Parameter	Description
<i>obj</i>	The object whose field value will be set. If the field is static, <i>obj</i> is ignored. For non-static fields, <i>obj</i> is required to be an instance of a class that inherits or declares the field.
<i>value</i>	A <code>System.Object</code> that contains the value to assign to the field contained by <i>obj</i> .

## 11 12 Description

13 Before setting the value, the system verifies that the user has access permission. If the  
14 user does not have access permission, a `System.FieldAccessException` is thrown.

## 15 Exceptions

Exception	Condition
<b>System.ArgumentException</b>	The field reflected by the current instance is declared neither directly in <i>obj</i> nor in any class from which <i>obj</i> derives.  <i>value</i> is not assignment-compatible with the type of the field reflected by the current instance.
<b>System.FieldAccessException</b>	The field reflected by the current instance is non-public, and the caller does not have permission to access non-public members.

<b>System.Reflection.TargetException</b>	The field reflected by the current instance is non-static, and <i>obj</i> is null.
--	--

1

2 **Permissions**

Permission	Description
<b>System.Security.Permissions.ReflectionPermission</b>	Requires permission to access non-public members of a type in loaded assemblies. See <code>System.Security.Permissions.ReflectionPermissionFlag.MemberAccess</code> .

3

4

# 1 FieldInfo.Attributes Property

```
2 [ILAsm]  
3 .property valuetype System.Reflection.FieldAttributes Attributes { public  
4 hidebysig virtual abstract specialname valuetype  
5 System.Reflection.FieldAttributes get_Attributes() }  
6 [C#]  
7 public abstract FieldAttributes Attributes { get; }
```

## 8 Summary

9 Gets the attributes of the field reflected by the current instance.

## 10 Property Value

11 A System.Reflection.FieldAttributes value that indicates the attributes of the field  
12 reflected by the current instance.

## 13 Behaviors

14 This property is read-only.

15

## 16 Usage

17 Use this property to determine the accessibility of the field reflected by the current  
18 instance. Also use this property to determine if the field reflected by the current instance  
19 can be set after it is initialized, is implemented in native code, is a literal, or has a  
20 special name.

21

## 22 Example

23 The following example demonstrates obtaining the attributes of two fields.

```
24 [C#]  
25  
26 using System;  
27 using System.Reflection;  
28  
29 class MyClass  
30 {  
31  
32     public int MyPublicInstanceField;  
33     private const int MyPrivateConstField = 10;  
34  
35 }  
36
```

```

1 class FieldAttributesExample
2 {
3
4     public static void Main()
5     {
6
7         Type t = (typeof(MyClass));
8         string str;
9         FieldInfo[] fiAry = t.GetFields( BindingFlags.Static |
10             BindingFlags.Instance | BindingFlags.Public |
11             BindingFlags.NonPublic | BindingFlags.DeclaredOnly );
12         foreach (FieldInfo fi in fiAry)
13         {
14             Console.WriteLine("Field {0} is: ", fi.Name);
15             str = ((fi.Attributes & FieldAttributes.Static) != 0) ?
16                 "Static": "Instance";
17             Console.Write(str + " ");
18             str = ((fi.Attributes & FieldAttributes.Public) != 0) ?
19                 "Public": "Not-Public";
20             Console.Write(str + " ");
21             str = ((fi.Attributes & FieldAttributes.Literal) != 0) ?
22                 "Literal": String.Empty;
23             Console.WriteLine(str);
24         }
25     }
26 }
27 }
28
29 }
30

```

31 The output is

32 Field MyPublicInstanceField is:

33

34

35 Instance Public

36

37

38

39 Field MyPrivateConstField is:

40

41

42

43

44

Static Not-Public Literal

# 1 FieldInfo.FieldType Property

```
2 [ILAsm]  
3 .property class System.Type FieldType { public hidebysig virtual abstract  
4 specialname class System.Type get_FieldType() }  
  
5 [C#]  
6 public abstract Type FieldType { get; }
```

## 7 Summary

8 Gets the type of the field reflected by the current instance.

## 9 Property Value

10 The `System.Type` of the field reflected by the current instance.

## 11 Description

12 This property is read-only.

13