C# Reflection

In C#, reflection is a *process to get metadata of a type at runtime*. The System.Reflection namespace contains required classes for reflection such as:

- Type
- MemberInfo
- ConstructorInfo
- MethodInfo
- FieldInfo
- PropertyInfo
- TypeInfo
- EventInfo
- Module
- Assembly
- AssemblyName
- Pointer etc.

The System.Reflection.Emit namespace contains classes to emit metadata.

C# Type class

C# Type class represents type declarations for class types, interface types, enumeration types, array types, value types etc. It is found in System namespace. It inherits System.Reflection.MemberInfo class.

C# Type Properties

A list of important properties of Type class are given below:

Property	Description
Assembly	Gets the Assembly for this type.
AssemblyQualifiedName	Gets the Assembly qualified name for this type.
Attributes	Gets the Attributes associated with the type.
BaseType	Gets the base or parent type.
FullName	Gets the fully qualified name of the type.
IsAbstract	is used to check if the type is Abstract.

IsArray	is used to check if the type is Array.
IsClass	is used to check if the type is Class.
IsEnum	is used to check if the type is Enum.
IsInterface	is used to check if the type is Interface.
IsNested	is used to check if the type is Nested.
IsPrimitive	is used to check if the type is Primitive.
IsPointer	is used to check if the type is Pointer.
IsNotPublic	is used to check if the type is not Public.
IsPublic	is used to check if the type is Public.
IsSealed	is used to check if the type is Sealed.
IsSerializable	is used to check if the type is Serializable.
MemberType	is used to check if the type is Member type of Nested type.
Module	Gets the module of the type.
Name	Gets the name of the type.
Namespace	Gets the namespace of the type.

C# Type Methods

A list of important methods of Type class are given below:

Method	Description
GetConstructors()	Returns all the public constructors for the Type.
GetConstructors(BindingFlags)	Returns all the constructors for the Type with specified BindingFlags.
GetFields()	Returns all the public fields for the Type.
GetFields(BindingFlags)	Returns all the public constructors for the Type with specified BindingFlags.
GetMembers()	Returns all the public members for the Type.

GetMembers(BindingFlags)	Returns all the members for the Type with specified BindingFlags.
GetMethods()	Returns all the public methods for the Type.
GetMethods(BindingFlags)	Returns all the methods for the Type with specified BindingFlags.
GetProperties()	Returns all the public properties for the Type.
GetProperties(BindingFlags)	Returns all the properties for the Type with specified BindingFlags.
GetType()	Gets the current Type.
GetType(String)	Gets the Type for the given name.

C# Reflection Example: Get Type

```
1. using System;
2. public class ReflectionExample
3. {
4.   public static void Main()
5.    {
6.       int a = 10;
7.       Type type = a.GetType();
8.       Console.WriteLine(type);
9.    }
10. }
```

Output:

```
System.Int32
```

C# Reflection Example: Get Assembly

```
1. using System;
2. using System.Reflection;
3. public class ReflectionExample
4. {
5.   public static void Main()
6.    {
7.       Type t = typeof(System.String);
8.       Console.WriteLine(t.Assembly);
9.    }
10. }
```

Output:

```
mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089
```

C# Reflection Example: Print Type Information

```
using System;
using System.Reflection;
public class ReflectionExample
{
    public static void Main()
    {
        Type t = typeof(System.String);
        Console.WriteLine(t.FullName);
        Console.WriteLine(t.BaseType);
        Console.WriteLine(t.IsClass);
        Console.WriteLine(t.IsEnum);
        Console.WriteLine(t.IsInterface);
    }
}
```

Output:

```
System.String
System.Object
true
false
false
```

C# Reflection Example: Print Constructors

```
using System;
using System.Reflection;
public class ReflectionExample
{
    public static void Main()
    {
        Type t = typeof(System.String);

        Console.WriteLine("Constructors of {0} type...", t);
        ConstructorInfo[] ci = t.GetConstructors(BindingFlags.Public | BindingFlags.Instance);
        foreach (ConstructorInfo c in ci)
        {
            Console.WriteLine(c);
        }
    }
}
```

Output:

```
Constructors of System.String type...
Void .ctor(Char*)
Void .ctor(Char*, Int32, Int32)
Void .ctor(SByte*)
Void .ctor(SByte*, Int32, Int32)
Void .ctor(SByte*, Int32, Int32, System.Text.Encoding)
Void .ctor(Char[], Int32, Int32)
Void .ctor(Char[])
Void .ctor(Char, Int32)
```

C# Reflection Example: Print Methods

```
1. using System;
2. using System.Reflection;
3. public class ReflectionExample
4. {
5.    public static void Main()
6.    {
7.       Type t = typeof(System.String);
8.
9.       Console.WriteLine("Methods of {0} type...", t);
10.      MethodInfo[] ci = t.GetMethods(BindingFlags.Public | BindingFlags.Instance);
11.      foreach (MethodInfo m in ci)
12.      {
13.         Console.WriteLine(m);
14.      }
15.   }
16. }
```

Output:

```
Methods of System.String type...
Boolean Equals(System.Object)
Boolean Equals(System.String)
Boolean Equals(System.String, System.StringComparison)
Char get_Chars(Int32)
Void copyTo(Int32, char[], Int32, Int32)
Char[] ToCharArray()
....
```

C# Reflection Example: Print Fields

```
1. using System;
2. using System.Reflection;
3. public class ReflectionExample
4. {
5.    public static void Main()
6.    {
7.       Type t = typeof(System.String);
```

```
8.
9.          Console.WriteLine("Fields of {0} type...", t);
10.         FieldInfo[] ci = t.GetFields(BindingFlags.Public | BindingFlags.Static | BindingFlags.NonPublic);
11.         foreach (FieldInfo f in ci)
12.         {
13.            Console.WriteLine(f);
14.         }
15.    }
16. }
```

Output:

```
Fields of System.String type...
System.String Empty
Int32 TrimHead
Int32 TrimTail
Int32 TrimBoth
Int32 charPtrAlignConst
Int32 alignConst
```