**Part 53 - C# Tutorial - Reflection**

**Reflection** is the ability of inspecting an assemblie's metadata at runtime.  It is used to find all types in an assembly and/or dynamically invoke methods in an assembly. This includes information about the type, properties, methods, and events of an object. With Reflection, we can dynamically create an instance of a type, bind the type to an existing object, or get the type from an existing object and invoke its methods or access its fields and properties.There are several uses of reflection.

**1.** When you drag and drop a button on a win forms or an asp.net application. The properties window uses reflection to show all the properties of the Button class. So,reflection is extensivley used by IDE or a UI designers.  
  
  
**2.** Late binding can be achieved by using reflection. You can use reflection to dynamically create an instance of a type, about which we don't have any information at compile time. So, reflection enables you to use code that is not available at compile time.  
  
  
**3.** Consider an example where we have two alternate implementations of an interface. You want to allow the user to pick one or the other using a config file. With reflection, you can simply read the name of the class whose implementation you want to use from the config file, and instantiate an instance of that class. This is another example for late binding using reflection.  
  
  
**So, in short reflection can be used for type discovery (i.e finding methods, properties, events, fields, constructors etc) and late binding.**  
  
  
In this session we will learn how to list a specifc class methods, properties, fields etc using reflection. All the classes and methods related to reflection are present in System.Reflection namespace.  
  
  
The Type class is the most importanct class.  
  
  
**Consider the Customer class example. This class has got**   
1. Two constructors  
2. Two auto implemeneted properties  
3. Two methods  
  
  
using System;  
using System.Reflection;  
namespace Pragim  
{  
    public class MainClass  
    {  
        private static void Main()  
        {  
            // Get the Type Using GetType() static method  
            Type T = Type.GetType("Pragim.Customer");  
            // Print the Type details  
            Console.WriteLine("Full Name = {0}",T.FullName);  
            Console.WriteLine("Just the Class Name = {0}",T.Name);  
            Console.WriteLine("Just the Namespace = {0}", T.Namespace);  
            Console.WriteLine();  
            // Print the list of Methods  
            Console.WriteLine("Methods in Customer Class");  
            MethodInfo[] methods = T.GetMethods();  
            foreach (MethodInfo method in methods)  
            {  
                // Print the Return type and the name of the method  
                Console.WriteLine(method.ReturnType.Name + " " + method.Name);  
            }  
            Console.WriteLine();  
            //  Print the Properties  
            Console.WriteLine("Properties in Customer Class");  
            PropertyInfo[] properties = T.GetProperties();  
            foreach (PropertyInfo property in properties)  
            {  
                // Print the property type and the name of the property  
                Console.WriteLine(property.PropertyType.Name + " " + property.Name);  
            }  
            Console.WriteLine();  
            //  Print the Constructors  
            Console.WriteLine("Constructors in Customer Class");  
            ConstructorInfo[] constructors = T.GetConstructors();

    foreach (ConstructorInfo constructor in constructors)  
            {  
                Console.WriteLine(constructor.ToString());  
            }  
        }  
    }  
    public class Customer  
    {  
        public int Id { get; set; }  
        public string Name { get; set; }  
  
  
        public Customer(int ID, string Name)  
        {  
            this.Id = ID;  
            this.Name = Name;  
        }  
  
  
        public Customer()  
        {  
            this.Id = -1;  
            this.Name = string.Empty;  
        }  
  
  
        public void PrintID()  
        {  
            Console.WriteLine("ID = {0}", this.Id);  
        }  
        public void PrintName()  
        {  
            Console.WriteLine("Name = {0}", this.Name);  
        }  
    }  
}  
  
  
In this example to get the type of customer class we have used **GetType()** static method defined on the **Type**class. We pass in the fully qualified name of the type including the namespace as a parameter to the GetType() method.  
Type T = Type.GetType("Pragim.Customer");  
  
  
**To get the type information we have the following 2 ways as well.**  
**Use typeof keyowrd**  
Type T = typeof(Customer);  
  
  
**Use GetType() on the instance of the customer class.**  
Customer C1 = new Customer();  
Type T = C1.GetType();  
  
  
To get the methods information, we use Type.GetMethods(), which returns MethodInfo[] array and along the same lines we use Type.GetProperties() to get properties information, but Type.GetProperties() returns PropertyInfo[] array.

[Email This](https://www.blogger.com/share-post.g?blogID=6082652835152798567&postID=439364570386432931&target=email)[BlogThis!](https://www.blogger.com/share-post.g?blogID=6082652835152798567&postID=439364570386432931&target=blog)[Share to Twitter](https://www.blogger.com/share-post.g?blogID=6082652835152798567&postID=439364570386432931&target=twitter)[Share to Facebook](https://www.blogger.com/share-post.g?blogID=6082652835152798567&postID=439364570386432931&target=facebook)[Share to Pinterest](https://www.blogger.com/share-post.g?blogID=6082652835152798567&postID=439364570386432931&target=pinterest)