

Override ToString() Method in C#

In C#, the ToString() method gives us a *String* representation of an object. It returns a string that represents the current object. We know that The Object class is the Superclass of all dot net types. That means, all the types in .NET Framework are inherited directly or indirectly from the Object class. Because of this inheritance, every type in .NET inherits the ToString() method from the Object class. If you go to the definition of Object class, then you will see that the ToString() method is defined as a Virtual Method which allows this method to be overridden in the child classes. Not only the ToString method but also you can override the Equals method which we will discuss in our next article.

```
namespace System
{
    ...public class Object
    {
        ...public Object()...

        ...public virtual string ToString()...
        ...public virtual bool Equals(object obj)...
        ...public static bool Equals(object objA, object objB)...
        ...public static bool ReferenceEquals(object objA, object objB)...
        ...public virtual int GetHashCode()...
        ...public extern Type GetType();

        ...~Object()...

        ...protected extern object MemberwiseClone();

        ...private void FieldSetter(string typeName, string fieldName, object val)...
        ...private void FieldGetter(string typeName, string fieldName, ref object val)...
        ...private FieldInfo GetFieldInfo(string typeName, string fieldName)...
    }
}
```

Note: Every type in .NET is implicitly inherited from the Object class and hence all the public and protected members of the object class (excluding private members) are inherited into the child class and by using the child class object we can access all the public and protected members of the object class in C#. For example, int is a primitive type and string a reference type and both of these two types are inherited from the Object class, and hence using the variable of int and string type, we can access all the public and protected members of the object class.

In other words, we can say that each and every class type (Reference Types) or struct type (Value Types) are directly or indirectly implicitly inherited from the Object class in C#. Therefore, every object in C# gets the ToString method, which returns a string representation of that object. So, the ToString() method returns a string that represents the current object.

For example, all variables of type int or float have the ToString method, which enables them to return their contents as a string. For a better understanding, please have a look at the following example. In the above example, the Number is an integer type variable and when we invoke the ToString() method on the Number object, it will give us the string representation of the integer 100.

```
using System;
namespace UnderstandingObjectClassMethods
{
    public class Program
    {
        public static void Main()
        {
            int Number = 100;
            Console.WriteLine(Number.ToString());
        }
    }
}
```

When you create a custom class or struct in C#, then you can override the ToString method in order to provide information about your type to the client. For example, if you have a complex type let's say Employee class as shown in the below example and when you call the ToString() method on the Employee object, then you will not get the output as expected. Hence we need to override the ToString() method, which is inherited from the Object class.

```
using System;
namespace UnderstandingObjectClassMethods
{
    public class Program
    {
        public static void Main()
        {
            Employee emp = new Employee();
            emp.FirstName = "Pranaya";
            emp.LastName = "Rout";
            Console.WriteLine(emp.ToString());
            Console.ReadKey();
        }
    }
}
```

```
public class Employee
{
    public string FirstName;
    public string LastName;
}
}
```

When you run the above code it will give you the below output. It is giving us the fully qualified name of the Employee type.

```
UnderstandingObjectClassMethods.Employee
```

Our requirement is when we call the ToString() method it should display the First Name and Last Name of the Employee object. To achieve this we need to override the ToString() Virtual method which is provided by the Object class in C#.

Overriding the ToString() Method in C#:

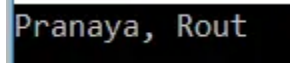
Please modify the code as shown below to override the ToString() method inside the Employee class. The point that you need to remember is the ToString method is defined as a Virtual Method inside the Object class and our custom Employee class is implicitly inherited from the Object class, and hence within this Employee class, we need to override the ToString method by using the override modifier which is shown in the below example.

```
using System;
namespace UnderstandingObjectClassMethods
{
    public class Program
    {
        public static void Main()
        {
            Employee emp = new Employee();
            emp.FirstName = "Pranaya";
            emp.LastName = "Rout";
            Console.WriteLine(emp.ToString());
            Console.ReadKey();
        }
    }

    public class Employee
    {
        public string FirstName;
        public string LastName;
```

```
//Overriding the Virtual ToString method of Object class
//Overriding the Virtual method using override modifier
public override string ToString()
{
    return FirstName + ", " + LastName;
}
}
```

Now run the application and you will see the First Name and Last Name of the employee as expected as shown below.

A screenshot of a console window or application output area. It shows the text "Pranaya, Rout" in a monospaced font. The text is white on a dark background, with a light blue vertical bar on the left side of the window.