C# Reflection API

The goal of this survey is to identify the perception of developers who use the C# Reflection API. It consists of three parts:

i. Participant Background (3 questions)
ii. The C# Reflection API (7 questions)
iii. Additional Comments (2 questions)

Answering the survey should take around 10-15 minutes of your time. All the data collected from the survey is anonymous. The results of the survey may be reported in academic publications. If you have any questions or concerns, please contact <blinded>.

Thanks,

<blinded>

* Required

Participant Background

In this section, we will ask questions about your profile.

1. **How many years of C# programming experience do you have?** *
   Mark only one oval.

   - ( ) I don't have experience developing C# applications *After the last question in this section, stop filling out this form.*
   - ( ) Less than one year
   - ( ) 1-3 years
   - ( ) 4-6 years
   - ( ) 7-10 years
   - ( ) More than 10 years

2. **Rate your background/knowledge about the C# Reflection API.** *
   Mark only one oval.

   - ( ) Not knowledgeable - I do not know anything about it *After the last question in this section, stop filling out this form.*
   - ( ) Somewhat knowledgeable - I have a vague idea about it
   - ( ) Knowledgeable - I am familiar with it
   - ( ) Very knowledgeable - I know all/most classes and methods of it

3. **How often do you need to use the C# Reflection API in your software applications?** *
Mark only one oval.

- ◯ Never *Stop filling out this form.*
- ◯ Sometimes - I need reflection for less than 33% of the software applications I develop
- ◯ Occasionally - I use reflection in more than 33% but less than 66% of the software applications I develop
- ◯ Frequently - I need reflection for more than 66% of the software applications I develop

The C# Reflection API

In this section, we present questions about the C# Reflection API.

Consider the following program:

```
using System;
using System.Reflection;

public class A
{
    public event Action Event { add { } remove { } }
}

public class B : A
{
    public static void Main(string[] args)
    {
        Type type = typeof(B);
        EventInfo eventInfo = type.GetEvent(nameof(A.Event));
        MemberInfo memberInfo = eventInfo.AddMethod;
        Console.WriteLine(memberInfo.ReflectedType);
    }
}
```

4. **What is the output of the above program? See documentation (https://goo.gl/TD3aMz).** *
Mark only one oval.

- ◯ Event
- ◯ B
- ◯ A
- ◯ Null
- ◯ Other: _____

Consider the following program:

```
using System;
using System.Linq;
using System.Reflection;

class A
{
    public static void Main(string[] args)
    {
        TypeInfo typeInfo = typeof(object[]).GetTypeInfo();
        ConstructorInfo constructor = typeInfo.DeclaredConstructors.ToArray()[0];
        Console.WriteLine(constructor.Invoke(new object[] { -1 }));
    }
}
```

5. **What is the output of the above program? See documentation (https://goo.gl/XJ1umS).** *
   Mark only one oval.

   - ◯ Null
   - ◯ System.Object[]
   - ◯ TargetInvocationException
   - ◯ OverflowException
   - ◯ Other: _____

Consider the following program:

```
using System;
using System.Reflection;

class A
{
    public static int Member1 = 0;
    public static void Member2() { }

    static void Main()
    {
        Type type = typeof(A);
        MemberInfo[] members = type.FindMembers(MemberTypes.All,
            BindingFlags.Public | BindingFlags.Static,
            Type.FilterAttribute, FieldAttributes.Static);
        for (int index = 0; index < members.Length; index++)
        {
            Console.WriteLine(members[index]);
        }
    }
}
```

6. **What is the output of the above program? See documentation (https://goo.gl/SCu3FR).** *
Mark only one oval.

- ◯ Null
- ◯ Void Member2(), System.Int32 Member1
- ◯ It throws an exception
- ◯ System.Int32 Member1, Void Member2()
- ◯ Other: _____

Consider the following program:

```
using System;
using System.Reflection;

public class A
{
    public void M<T>() { }
}

public class B : A
{
    public static void Main(string[] args)
    {
        Type t = typeof(B);
        MethodInfo method = t.GetMethod("M");
        Console.WriteLine(method.MakeGenericMethod(typeof(int)));
    }
}
```

7. **What is the output of the above program? See documentation (https://goo.gl/uSNYdb).** *
Mark only one oval.

- ◯ Null
- ◯ Void M[Int32]()
- ◯ Void M[T]()
- ◯ It throws an exception
- ◯ Other: _____

Consider the following program:

```
using System;
using System.Reflection;
public class A
{
    public static void Main()
    {
        Assembly assem = typeof(A).Assembly;
        A a = (A)assem.CreateInstance(" ");
        if (a == null)
        {
            Console.WriteLine("Null");
        }
        else
        {
            Console.WriteLine(a);
        }
    }
}
```

8. **What is the output of the above program? See documentation (https://goo.gl/TVyr4q).** *
Mark only one oval.

- A
- System.Object
- Null
- It throws an exception
- Other: _____

Consider the following program:

```
using System;
using System.Reflection;
public class A
{
    public int Field;
    public static void Main()
    {
        Type type = typeof(A);
        FieldInfo field = type.GetField("");
        if (field == null)
        {
            Console.WriteLine("Null");
        }
        else
        {
            Console.WriteLine(field);
        }
    }
}
```

9. **What is the output of the above program? See documentation (https://goo.gl/1CvrRp)** *
Mark only one oval.

- ◯ Null
- ◯ System.Object
- ◯ It throws an exception
- ◯ Field
- ◯ Other: _____

Consider the following program:

```
using System;
using System.Reflection;
public class A
{
    public void M(int p) {}
}
public class B : A
{
    public void M(string p) {}
    public static void Main()
    {
        Type type = typeof(B);
        MemberInfo[] membersArray = type.GetMember("M");
        for (int index = 0; index < membersArray.Length; index++)
        {
            Console.WriteLine(membersArray[index].ToString());
        }
    }
}
```

10. **What is the output of the above program? See documentation (https://goo.gl/WcEEx1).** *
Mark only one oval.

- ◯ Void M(Int32)
- ◯ Void M(System.String) and Void M(Int32)
- ◯ Void M(System.String)
- ◯ It throws an exception
- ◯ Other: _____

Additional Comments

11. **Please, let us know if you have any additional comments about the C# Reflection API.**

12. **If you would like to receive the results of our survey, please leave your email address.**