

# Reflection and Attributes



**SoftUni Team**  
Technical Trainers



**SoftUni**



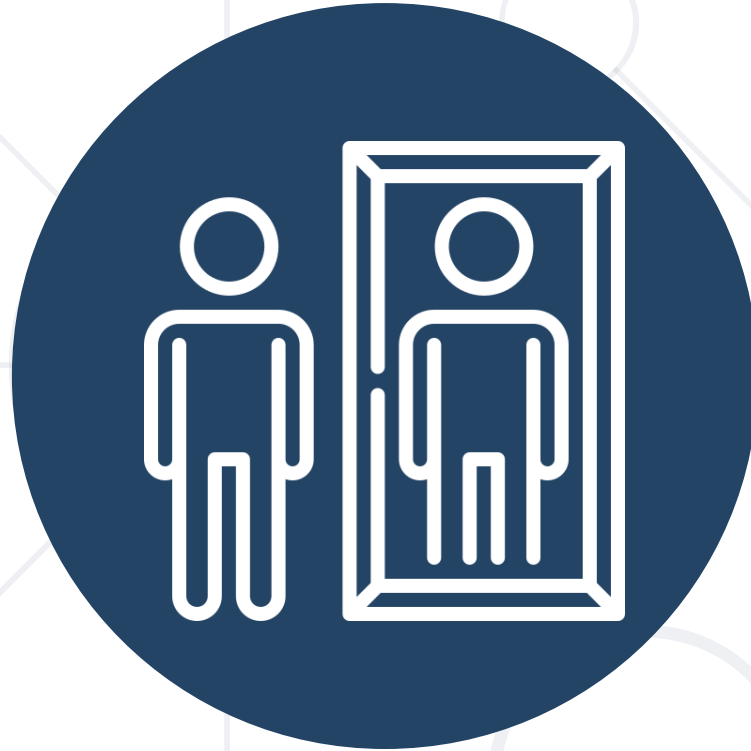
**Software University**

<https://softuni.bg>

- Reflection - What? Why? Where? When?
- Reflection API
  - Type Class
  - Reflecting Fields
  - Reflecting Constructors
  - Reflecting Methods
- Attributes
  - Applying Attributes to Code Elements
  - Built-in Attributes
  - Defining Attributes

[sli.do](https://sli.do)

**#csharp-advanced**



**What? Why? Where? When?**

Reflection

# What is Metaprogramming?

- **Programming technique**, in which computer programs have the ability to treat other **programs as their data**
- Programs can be designed to:
  - **Read**
  - **Generate**
  - **Analyze**
  - **Transform**
  - **Modify itself** while **running**



# What is Reflection?

- The ability of a programming language to be its **own metalanguage**
- Programs can examine information about **themselves**



# When to Use Reflection?

- Whenever we want:
  - Code to become more **extendible** (e.g. plugins)
  - To **reduce code** length significantly (e.g. mapping)
  - Dynamic object **initialization** (e.g. IoC containers)
  - **Assembly** information at run time (e.g. ASP.NET Core)
  - Examine other **programs** (e.g. unit testing)



# When Not to Use Reflection?

- If it is **possible** to **perform** an operation **without** using **reflection**, then it is preferable to **avoid using it**
- Cons from using Reflection
  - **Performance** overhead
  - **Security** restrictions
  - Exposure of **internals**





# Reflecting Class and Members

Reflection API

- Primary way to access **metadata**
- Obtained at **compile time**, if you know its **name**:

```
Type myType = typeof(ClassName);
```

- Can be obtained at **runtime**, if the name is **unknown**:

```
Type myType = Type.GetType("Namespace.ClassName");
```

You need fully qualified class name as string

- Get the type of an instance

```
obj.GetType();
```

- Obtain Class name
  - Fully qualified class name - `Type.FullName`
  - Class name without the namespace - `Type.Name`

```
string fullName = typeof(SomeClass).FullName;
```

```
string simpleName = typeof(SomeClass).Name;
```

- Obtain **base type**

```
Type baseType = testClass.BaseType;
```

- Obtain **interfaces**

```
Type[] interfaces = testClass.GetInterfaces();
```

- All the **interfaces that the class implements** are returned
  - Even interfaces from **base classes**

- **Activator.CreateInstance**
  - Creates an instance of a type by invoking the constructor that **matches** the specified **arguments**

```
var sbType = Type.GetType("System.Text.StringBuilder");  
StringBuilder sbInstance =  
    (StringBuilder) Activator.CreateInstance(sbType);  
StringBuilder sbInstCapacity = (StringBuilder)Activator  
    .CreateInstance(sbType, new object[] { 10 });
```

- Obtain public fields

```
FieldInfo field = type.GetField("name");  
FieldInfo[] publicFields = type.GetFields();
```

- Obtain all fields

```
FieldInfo[] allFields = type.GetFields(  
    BindingFlags.Static |  
    BindingFlags.Instance |  
    BindingFlags.Public |  
    BindingFlags.NonPublic);
```

- The **BindingFlags** enum specifies what kinds of types we are looking up

```
FieldInfo[] allFields =  
    type.GetFields(BindingFlags.NonPublic);
```

- Can be combined with bitwise OR (| operator):

```
FieldInfo[] allFields = type.GetFields(  
    BindingFlags.Public |  
    BindingFlags.NonPublic);
```

Returns both public  
and nonpublic fields

- Get **public** field **name** and **type**

```
FieldInfo field = type.GetField("fieldName");  
string fieldName = field.Name;  
Type fieldType = field.FieldType;
```

- Use **BindingFlags** to specify access modifiers, if the field is not public, otherwise **GetField** returns **null**



# Changing a Field's State

```
Type testType = typeof(Test);  
Test testInstance =  
    (Test) Activator.CreateInstance(testType);  
FieldInfo field = testType.GetField("testInt");  
  
field.SetValue(testInstance, 5);  
int fieldValue =  
    (int)field.GetValue(testInstance);
```

Changes the  
object's state

- Each modifier is a **flag bit** that is either set or cleared
- Check **access modifier** of a **member** of the class

```
field.IsPrivate      // private
field.IsPublic       // public
field.IsNonPublic    // everything but public
field.IsFamily       // protected
field.IsAssembly     // internal
```

- Obtain **constructors**

```
ConstructorInfo[] publicCtors =  
    type.GetConstructors();
```

- Obtain **all non static constructors**

```
ConstructorInfo[] allNonStaticCtors =  
    type.GetConstructors(  
        BindingFlags.Instance |  
        BindingFlags.Public |  
        BindingFlags.NonPublic);
```

# Reflect Constructors(2)

- Obtain a certain constructor

```
ConstructorInfo constructor =  
    type.GetConstructor(new Type[] parametersType);
```

- Get constructor parameters

```
Type[] parameterTypes =  
    constructor.GetParameters();
```

- Instantiating objects using a specific constructor

```
StringBuilder builder =  
    (StringBuilder)constructor.Invoke(  
        new object[] { "gosho", 5 });
```

Supply positional parameters  
in an object array

- Obtain all **public** methods

```
MethodInfo[] publicMethods = sbType.GetMethods();
```

- Obtain a **certain** method

```
MethodInfo appendMethod =  
    sbType.GetMethod("Append");  
MethodInfo overloadMethod = sbType.GetMethod(  
    "Append", new [] { typeof(array) });
```

- Obtain method **parameters** and **return type**

```
ParameterInfo[] appendParameters =  
    appendMethod.GetParameters();  
Type returnType = appendMethod.ReturnType;
```

- **Invoke** methods

```
appendMethod.Invoke(builder, new object[] { "hi!" });
```

Target object  
instance

Parameters for  
the method




# Data about Data

Attributes

# Attributes

- **Data holding** class
- **Describes** parts of your code
- Applied to:
  - **Classes, Fields, Methods**, etc.



```
[Obsolete]
public void DeprecatedMethod
{
    Console.WriteLine("Deprecated!");
}
```



- Generate **compiler messages** or **errors**

**[Obsolete]**

```
public enum Coin // Enum 'Coin' is obsolete
```

- Tools, which rely on attributes:
  - **Code generation** tools
  - **Documentation generation** tools
  - **Testing** Frameworks
- Runtime - **ORM, Serialization** etc.

- Attribute's name is surrounded by **square brackets: []**
  - Placed before their target declaration

```
[Flags] // System.FlagsAttribute
public enum FileAccess
{
    Read = 1,
    Write = 2,
    ReadWrite = Read | Write
}
```

- **[Flags]** attribute indicates that the enum type can be treated like a set of bit flags, stored as a single integer

- Attributes can accept **parameters** for their constructors and public properties

```
[DllImport("user32.dll", EntryPoint="MessageBox")]  
public static extern int ShowMessageBox(int hWnd,  
    string text, string caption, int type);  
...  
ShowMessageBox(0, "Some text", "Some caption", 0);
```

- The [DllImport] attribute refers to:
  - **System.Runtime.InteropServices.DllImportAttribute**
  - **"user32.dll"** is passed to the constructor
  - **"MessageBox"** value is assigned to **EntryPoint**

- Must **inherit** the **System.Attribute** class
- Their **names** must end with "**Attribute**"
- Possible **targets** must be defined via **[AttributeUsage]**
- Can define **constructors** with parameters
- Can define public **fields** and **properties**

# Problem: Create Attribute

- Create an attribute **Author** with a **string** element called **name** that:
  - Can be used over **classes and methods**
  - Allow multiple attributes of same type

```
[Author("Victor")]
public class StartUp
{
    [Author("Georg")]
    static void Main(string[] args)
    { ... }
}
```

Check your solution here: <https://judge.softuni.bg/Contests/1520/Reflection-and-Attributes-Lab>

# Solution: Create Attribute

```
[AttributeUsage(AttributeTargets.Class |  
                AttributeTargets.Method,  
                AllowMultiple = true)]  
public class AuthorAttribute : Attribute  
{  
    public AuthorAttribute(string name)  
    {  
        this.Name = name;  
    }  
  
    public string Name { get; set; }  
}
```

# Problem: Coding Tracker

- Create a class **Tracker** with a method:
  - **void PrintMethodsByAuthor()**
- Print to the console authors for all methods
  - Use **SoftUni** attribute and **reflection**

Check your solution here: <https://judge.softuni.bg/Contests/1520/Reflection-and-Attributes-Lab>

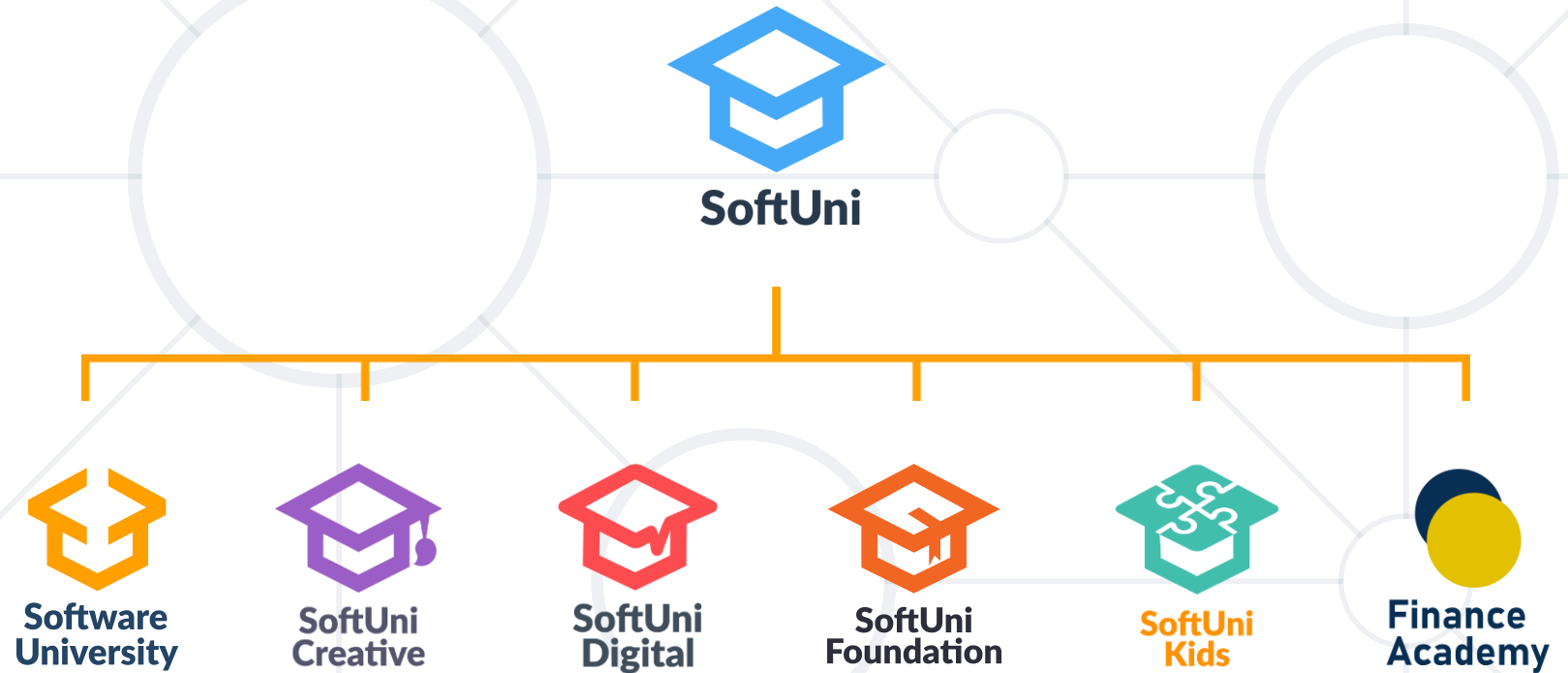
```
var type = typeof(Startup);
var methods =
    type.GetMethods(
        BindingFlags.Instance | BindingFlags.Public |
        BindingFlags.Static);
foreach (var method in methods) {
    if(method.CustomAttributes
        .Any(n => n.AttributeType == typeof(AuthorAttribute))) {
        var attributes = method.GetCustomAttributes(false);
        foreach(AuthorAttribute attr in attributes){
            Console.WriteLine("{0} iw written by {1}",
                               method.Name, attr.Name);
        }
    }
}
// Add the missing brackets
```



- **Reflection:**
  - Allows us to get **information about types**
  - Allows us to dynamically **call methods**, **get/set** values, etc.
- **Attributes** allow adding metadata in classes / types / etc.
  - Built-in attributes
  - Custom attributes
  - Can be accessed at runtime



# Questions?



# SoftUni Diamond Partners

**SUPER  
HOSTING  
.BG**



**Coca-Cola HBC  
Bulgaria**

 **Flutter**<sup>TM</sup>  
International

**INDEAVR**  
Serving the high achievers



**AMBITIONED**

 **DRAFT  
KINGS**



**BOSCH**

 **Postbank**  
*Решения за твоето утре*

 **PHAR  
VISION**



**SmartIT**

**DXC**  
TECHNOLOGY

**createX**

- Software University – High-Quality Education, Profession and Job for Software Developers

- [softuni.bg](http://softuni.bg), [about.softuni.bg](http://about.softuni.bg)

- Software University Foundation

- [softuni.foundation](http://softuni.foundation)

- Software University @ Facebook

- [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)

- Software University Forums

- [forum.softuni.bg](http://forum.softuni.bg)



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg>
- © Software University – <https://softuni.bg>

