Examples of useful Roslyn Source Code Generator

About this book

Content of the book

You will find in this book code examples about >10 Roslyn Source Code Generator (RSCG) that can be usefull for you. That means, you will write more elegant and concise code - even if the generators code is not always nice to look.

Are those examples ready for production ?

I have done due diligence to test the RSCG that I have show to you here. However, I cannot guarantee that will fit your code . That means that you can test it for your case and, because all are open source on Github.com , you can contribute to improve them ;-)

How to read this book

For each chapter, you will find

1. Name of the RSCG and link to the NuGet package / GitHub repository
2. What the RSCG can do
3. What will be the initial code
4. How to use the Code generated by RSCG
5. Code Generated by RSCG
6. Link to the downloadable code to practice

I have a suggestion for a new RSCG that is worth mentioning in this book . What can I do ?

Please send me an email to ignatandrei@yahoo.com

I want to make a RSCG that will be useful. How can I do ?

In the introduction I have put the links to get you started with RSCG .

And, if you bought this book from Amazon , you are entitled to have 1 hour free of consultancy with me . I can help you make one.

Introduction

What is a Roslyn Source Code Generator ?

A Roslyn Source Code Generator (RSCG) is a program that generates code in the compile time, based on the previous source code and/or another data . This new source code is added to the compilation and compile with the previous source code.

How can I make a Roslyn Source Code Generator ?

For creating the RSCG you will simply create a .NET Standard 2.0 project,add those 2 references

```
    <PackageReference Include="Microsoft.CodeAnalysis.Analyzers"
Version="3.3.1" PrivateAssets="all" />
    <PackageReference Include="Microsoft.CodeAnalysis.CSharp" Version="3.8.0"
/>
```

and start implementing

```
public interface ISourceGenerator
{
    void Initialize(GeneratorInitializationContext context);
    void Execute(GeneratorExecutionContext context);
}
```

Start from examples at https://github.com/dotnet/roslyn-sdk/tree/main/samples/CSharp/SourceGenerators
Also, you can read the source code for the RSCG presented in this book.

Show me some code for RSCG

Start read

https://github.com/dotnet/roslyn/blob/main/docs/features/source-generators.md

and

https://github.com/dotnet/roslyn/blob/main/docs/features/source-generators.cookbook.md .

After that , you can play with the examples from https://github.com/dotnet/roslyn-sdk/tree/main/samples/CSharp/SourceGenerators or from https://sourcegen.dev/ (see AutoNotify in the dropdown)

How the RSCG can help me to write faster / better the code ?

Glad that you asked. You can see in action a RSCG for automatically generating code for automating testing (see DynamicMocking) , parsing enum (see Enum) , generating controllers actions from a interface (SkinnyControllers), currying functions and many more. In this book you will find more than 10 examples of some RSCG that can help you. Also, you can find the source code of the examples at https://github.com/ignatandrei/RSCG_Examples.

RSCG number 1 : ThisAssembly

Nuget :
https://www.nuget.org/packages/ThisAssembly

link : https://www.clarius.org/ThisAssembly/

author :Daniel Cazzulino

What can do

The ThisAssembly.Info allows you access to the Assembly Information as constants, instead of going to reflection each time. I found useful to see the assembly version right away in any project that I have.

The code that you start with is

```
<PropertyGroup>

<Version>2021.2.15.800</Version>

</PropertyGroup>
```

The code that you will use is

```
var strVersion=ThisAssembly.Info.Version;

Console.WriteLine(strVersion);
```

The code that is generated is

```
        /// <summary>

        /// Provides access to the current assembly information as pure
constants,

        ///  without requiring reflection.

        /// </summary>
```

```
partial class ThisAssembly

{

    /// <summary>

    /// Gets the AssemblyInfo attributes.

    /// </summary>

    [GeneratedCode("ThisAssembly.AssemblyInfo", "1.0.0")]

    [CompilerGenerated]

    public static partial class Info

    {

        public const string Company = @"RSCG_Version";


        public const string Configuration = @"Debug";


        public const string FileVersion = @"2021.2.15.800";


        public const string InformationalVersion = @"2021.2.15.800";


        public const string Product = @"RSCG_Version";


        public const string Title = @"RSCG_Version";


        public const string Version = @"2021.2.15.800";


    }
```

```
        }
```

Example Code: https://github.com/ignatandrei/RSCG_Examples/tree/main/ApplicationVersion (https://github.com/ignatandrei/RSCG_Examples/tree/main/ApplicationVersion)

All Generators: https://github.com/ignatandrei/RSCG_Examples/ (https://github.com/ignatandrei/RSCG_Examples/)

RSCG number 2 : Enum

Nuget :
https://www.nuget.org/packages/AOPMethodsCommon/
https://www.nuget.org/packages/AOPMethodsGenerator/

link : http://msprogrammer.serviciipeweb.ro/category/roslyn/

author :Andrei Ignat

What can do

This will generate code to fast parsing a int or a string to an enum

The code that you start with is

```
[AutoEnum(template = EnumMethod.GenerateExtensionCode)]

public enum MathematicalOperation

{

None=0,

Add=1,

Multiplication=2

}
```

The code that you will use is

```
var fromInt = enumMathematicalOperation.ParseExactMathematicalOperation(1);

var fromString =
enumMathematicalOperation.ParseExactMathematicalOperation("add");

Console.WriteLine(fromInt + "-"+fromString);
```

The code that is generated is

```
     [GeneratedCode("AOPMethods", "")]

     [CompilerGenerated]

     public  static partial class enumMathematicalOperation{

      /*

       public static int idMathematicalOperation(){

       System.Diagnostics.Debugger.Break();

       return 1;

       }

       */

       public static RSCG_Enum.MathematicalOperation
ParseExactMathematicalOperation(this long value,
RSCG_Enum.MathematicalOperation? defaultValue = null){

               if(0 == value)

                   return RSCG_Enum.MathematicalOperation.None;

                       if(1 == value)

                   return RSCG_Enum.MathematicalOperation.Add;

                       if(2 == value)

                   return RSCG_Enum.MathematicalOperation.Multiplication;



           if(defaultValue != null)

               return defaultValue.Value;



           throw new ArgumentException("cannot find " + value +" for
RSCG_Enum.MathematicalOperation  ");

       }
```

```
        public static RSCG_Enum.MathematicalOperation
ParseExactMathematicalOperation(this string value,
RSCG_Enum.MathematicalOperation? defaultValue = null){

            //trying to see if it is a value inside

            //if(!string.IsNullOrWhiteSpace)

            if(long.TryParse(value, out long valueParsed)){

                return ParseExactMathematicalOperation(valueParsed);

            }



                if(0==string.Compare("None" , value,
StringComparison.InvariantCultureIgnoreCase))

                    return RSCG_Enum.MathematicalOperation.None;

                        if(0==string.Compare("Add" , value,
StringComparison.InvariantCultureIgnoreCase))

                    return RSCG_Enum.MathematicalOperation.Add;

                        if(0==string.Compare("Multiplication" , value,
StringComparison.InvariantCultureIgnoreCase))

                    return RSCG_Enum.MathematicalOperation.Multiplication;




            if(defaultValue != null)

                return defaultValue.Value

            throw new ArgumentException("cannot find " + value +" for
RSCG_Enum.MathematicalOperation  ");

        }

        /*
```

```
        */


    }
```

Example Code: https://github.com/ignatandrei/RSCG_Examples/tree/main/Enum (https://github.com/ignatandrei/RSCG_Examples/tree/main/Enum)

All Generators: https://github.com/ignatandrei/RSCG_Examples/ (https://github.com/ignatandrei/RSCG_Examples/)

RSCG number 3 : JsonByExampleGenerator

Nuget : https://www.nuget.org/packages/JsonByExampleGenerator/

link : https://github.com/hermanussen/JsonByExampleGenerator/

author :Robin Hermanussen

What can do

This will generate C# classes from json files.

The code that you start with is

```
{
"FirstName": "Andrei",
"LastName": "Ignat",
"Blog": "http://msprogrammer.serviciipeweb.ro/"
```

The code that you will use is

```
var p1 = new Person();
p1.Blog = "http://msprogrammer.serviciipeweb.ro/";
var config = new ConfigurationBuilder()
  .AddJsonFile("persons.json")
  .Build();

var p = config.Get<Person>();
var p2 = Person.FromConfig(config);
```

The code that is generated is

```
    [DataContract(Name = "Person", Namespace = "JsonToClass.Json.Persons")]
    public partial class Person
    {
    [DataMember(Name = "FirstName", EmitDefaultValue = false, Order = 0)]
    public string FirstName { get; set; }
    [DataMember(Name = "LastName", EmitDefaultValue = false, Order = 1)]
    public string LastName { get; set; }
    [DataMember(Name = "Blog", EmitDefaultValue = false, Order = 2)]
    public string Blog { get; set; }

    public static Person FromConfig([System.Diagnostics.CodeAnalysis.NotNull]
IConfiguration config)
    {
    return config.Get<Person>();
    }
    }
```

Example Code: https://github.com/ignatandrei/RSCG_Examples/tree/main/JsonToClass (https://github.com/ignatandrei/RSCG_Examples/tree/main/JsonToClass)

All Generators: https://github.com/ignatandrei/RSCG_Examples/ (https://github.com/ignatandrei/RSCG_Examples/)

RSCG number 4 : CopyConstructor + Deconstructor

Nuget :
https://www.nuget.org/packages/AOPMethodsCommon/
https://www.nuget.org/packages/AOPMethodsGenerator/

link : http://msprogrammer.serviciipeweb.ro/category/roslyn/

author :Andrei Ignat

What can do

This will generate code for a POCO to generate copy constructor and deconstructor

The code that you start with is

```
    [AutoMethods(template = TemplateMethod.CustomTemplateFile,
CustomTemplateFileName = "CopyConstructorDestructor.txt")]

    partial class Person

    {

       public string FirstName { get; set; }

       public string LastName { get; set; }

    }
```

The code that you will use is

```
var pOldPerson = new Person();

pOldPerson.FirstName = "Andrei";

pOldPerson.LastName = "Ignat";

var newPerson = new Person(pOldPerson);

Console.WriteLine(newPerson.FirstName);

var (_, last) = newPerson;

Console.WriteLine(last);
```

The code that is generated is

```
public Person (){
   OnConstructor();
}
public Person(IPerson other):base(){
     BeforeCopyConstructor(other);
     CopyPropertiesFrom(other);
     AfterCopyConstructor(other);

}
public void CopyPropertiesFrom(IPerson other){

     this.FirstName = other.FirstName;
     this.LastName = other.LastName;
}




 public void Deconstruct( out string FirstName, out string LastName)
 {
     FirstName = this.FirstName;
     LastName = this.LastName;
 }
```

Examples of useful Roslyn Source Code Generator

Example Code: https://github.com/ignatandrei/RSCG_Examples/tree/main/CopyConstructor (https://github.com/ignatandrei/RSCG_Examples/tree/main/CopyConstructor)

All Generators: https://github.com/ignatandrei/RSCG_Examples/ (https://github.com/ignatandrei/RSCG_Examples/)

RSCG number 5 : GeneratedMapper

Nuget :
https://www.nuget.org/packages/GeneratedMapper/

link : https://github.com/ThomasBleijendaal/GeneratedMapper

author :Thomas Bleijendaal

What can do

AutoMapping from a POCO to a DTO. Lots of customizations

The code that you start with is

```
public class Department

    {

        public int ID { get; set; }

        public string Name { get; set; }



        public List<string> Employees { get; set; }

    }



    [IgnoreInTarget("Employees")]

    [MapFrom(typeof(Department))]

    public class DepartmentDTO

    {

        public int ID { get; set; }

        public string Name{get; set;}
```

```
        [MapWith("Employees",typeof(ResolverLength))]
        public int EmployeesNr { get; set; }


    }
    public class ResolverLength
    {
        public int Resolve(List<string> input)
        {
            return ((input?.Count) ?? 0);
        }
    }
```

The code that you will use is

```
static void Main(string[] args)
{
    var dep = new Department();
    dep.Name = "IT";
    dep.ID = 1;
    dep.Employees = new List<string>();
    dep.Employees.Add("Andrei");
    var dto = dep.MapToDepartmentDTO();
    Console.WriteLine(dto.Name+"=>"+ dto.EmployeesNr);
}
```

The code that is generated is

```
    namespace DTOMapper

    {

        public static partial class DepartmentMapToExtensions

        {

            public static DTOMapper.DepartmentDTO MapToDepartmentDTO(this
DTOMapper.Department self)

            {

                if (self is null)

                {

                    throw new ArgumentNullException(nameof(self),
"DTOMapper.Department -> DTOMapper.DepartmentDTO: Source is null.");

                }



                var resolverLength = new DTOMapper.ResolverLength();



                var target = new DTOMapper.DepartmentDTO

                {

                    ID = self.ID,

                    Name = (self.Name ?? throw new
GeneratedMapper.Exceptions.PropertyNullException("DTOMapper.Department ->
DTOMapper.DepartmentDTO: Property Name is null.")),

                    EmployeesNr = resolverLength.Resolve((self.Employees ??
throw new
GeneratedMapper.Exceptions.PropertyNullException("DTOMapper.Department ->
DTOMapper.DepartmentDTO: Property Employees is null."))),

                };
```

```
            return target;
        }
    }
}
```

Example Code: https://github.com/ignatandrei/RSCG_Examples/tree/main/DTOMapper (https://github.com/ignatandrei/RSCG_Examples/tree/main/DTOMapper)

All Generators: https://github.com/ignatandrei/RSCG_Examples/ (https://github.com/ignatandrei/RSCG_Examples/)

RSCG number 6 : Skinny Controllers

Nuget :
https://www.nuget.org/packages/SkinnyControllersCommon/
https://www.nuget.org/packages/SkinnyControllersGenerator/

link : http://msprogrammer.serviciipeweb.ro/category/roslyn/

author :Andrei Ignat

What can do

This will generate code for WebAPI for each method of a field in the controller

The code that you start with is

```
public class PersonRepository

{

    public async Task<Person> Get(int id)

    {

        await Task.Delay(1000);

        return new Person()

        {

            ID = id,

            Name = "Andrei " + id

        };

    }



    //add more functions here to make the demo

}
```

The code that you will use is

```
    [AutoActions(template = TemplateIndicator.AllPostWithRecord,  FieldsName =
new[] { "*" }, ExcludeFields = new[] { "_logger" })]

    [ApiController]

    [Route("[controller]/[action]")]

    public partial class PersonController : ControllerBase

    {

        private readonly PersonRepository pr;

        private readonly ILogger<PersonController> _logger;



        public PersonController(PersonRepository pr, ILogger<PersonController>
logger)

        {

            this.pr = pr;

            _logger = logger;

        }



    }
```

The code that is generated is

```
    [HttpPost]

    public System.Threading.Tasks.Task<AOPSkinnyController.Classes.Person> Get(
recGet_143266108 data ){



            return



        pr.Get(data.id);



    }
```

Example Code: https://github.com/ignatandrei/RSCG_Examples/tree/main/SkinnyControllers (https://github.com/ignatandrei/RSCG_Examples/tree/main/SkinnyControllers)

All Generators: https://github.com/ignatandrei/RSCG_Examples/ (https://github.com/ignatandrei/RSCG_Examples/)

RSCG number 7 : data-builder-generator

Nuget :
https://www.nuget.org/packages/DasMulli.DataBuilderGenerator/

link : https://github.com/dasMulli/data-builder-generator

author :Martin Andreas Ulrich

What can do

Implements the Builder Design pattern for any class. Useful , at least, for test projects

The code that you start with is

```
[GenerateDataBuilder]
public class Person
{
     public string FirstName { get; set; }
     public string? MiddleNames { get; set; }
     public string LastName { get; set; }

}
```

The code that you will use is

```
    var pOld = new Person();
    pOld.FirstName = "Andrei";
    pOld.LastName = "Ignat";
    pOld.MiddleNames = "G";
    var build = new
PersonBuilder(pOld).WithoutMiddleNames().WithFirstName("Florin");
    var pNew = build.Build();
    Console.WriteLine(pNew.FirstName);
```

The code that is generated is

```
    public partial class PersonBuilder
        {
            private string? _firstName;
            private string? _middleNames;
            private string? _lastName;
            public PersonBuilder()
            {
            }

            public PersonBuilder(PersonBuilder otherBuilder)
            {
                _firstName = otherBuilder._firstName;
                _middleNames = otherBuilder._middleNames;
```

```
            _lastName = otherBuilder._lastName;
        }

        public PersonBuilder(Person existingInstance)
        {
            _firstName = existingInstance.FirstName;
            _middleNames = existingInstance.MiddleNames;
            _lastName = existingInstance.LastName;
        }

        public PersonBuilder WithFirstName(string firstName)
        {
            var mutatedBuilder = new PersonBuilder(this);
            mutatedBuilder._firstName = firstName;
            return mutatedBuilder;
        }

        public PersonBuilder WithMiddleNames(string? middleNames)
        {
            var mutatedBuilder = new PersonBuilder(this);
            mutatedBuilder._middleNames = middleNames;
            return mutatedBuilder;
        }
```

```
public PersonBuilder WithoutMiddleNames()
{
    var mutatedBuilder = new PersonBuilder(this);
    mutatedBuilder._middleNames = null;
    return mutatedBuilder;
}

public PersonBuilder WithLastName(string lastName)
{
    var mutatedBuilder = new PersonBuilder(this);
    mutatedBuilder._lastName = lastName;
    return mutatedBuilder;
}

public Person Build()
{
    var instance = new Person();
    if (!(_firstName is null))
        instance.FirstName = _firstName;
    if (!(_middleNames is null))
        instance.MiddleNames = _middleNames;
    if (!(_lastName is null))
        instance.LastName = _lastName;
    return instance;
```

```
            }

        }
```

Example Code: https://github.com/ignatandrei/RSCG_Examples/tree/main/DP_Builder (https://github.com/ignatandrei/RSCG_Examples/tree/main/DP_Builder)

All Generators: https://github.com/ignatandrei/RSCG_Examples/ (https://github.com/ignatandrei/RSCG_Examples/)

RSCG number 8 : Metadata from object

Nuget :
https://www.nuget.org/packages/AOPMethodsCommon/
https://www.nuget.org/packages/AOPMethodsGenerator/

link : http://msprogrammer.serviciipeweb.ro/category/roslyn/

author :Andrei Ignat

What can do

This will generate code to retrieve the values of properties directly, not by reflection

The code that you start with is

```
    [AutoMethods(template = TemplateMethod.CustomTemplateFile,
CustomTemplateFileName = "GenerateFromPOCO.txt")]

    public partial class Person

    {

        public string FirstName { get; set; }

        public string LastName { get; set; }

    }
```

The code that you will use is

```
var p = new Person();
p.FirstName = "Andrei";
p.LastName = "Ignat";
var last = p.ValueProperty(Person_EnumProps.LastName);
var first = p.ValueProperty("FirstName");

Console.WriteLine(last + " "+first);
```

The code that is generated is

```
public enum Person_EnumProps{
    None
    ,FirstName // Public
    ,LastName // Public
}
partial class Person{
    public object ValueProperty(Person_EnumProps val){
        if(val == Person_EnumProps.FirstName) {
            return this.FirstName;
        }
        if(val == Person_EnumProps.LastName) {
            return this.LastName;
        }
    throw new ArgumentException("cannot find "+ val);
```

```
        }

        public object ValueProperty(string val){


if(string.Compare("FirstName",val,StringComparison.CurrentCultureIgnoreCase)==0
) {

                return this.FirstName;

            }


if(string.Compare("LastName",val,StringComparison.CurrentCultureIgnoreCase)==0)
{

                return this.LastName;

            }

        throw new ArgumentException("cannot find "+ val);

        }

    }
```

Example Code: https://github.com/ignatandrei/RSCG_Examples/tree/main/MetadataFromObject (https://github.com/ignatandrei/RSCG_Examples/tree/main/MetadataFromObject)

All Generators: https://github.com/ignatandrei/RSCG_Examples/ (https://github.com/ignatandrei/RSCG_Examples/)

RSCG number 9 : MockSourceGenerator

Nuget :
https://www.nuget.org/packages/MockSourceGenerator/

link : https://github.com/hermanussen/MockSourceGenerator/

author :Robin Hermanussen

What can do

This will generate Mock classes directly for any interface - with your implementation.

The code that you start with is

```
public interface IMatOps

{

    public int Add(int a, int b);



    public int Division(int a, int b);

}
```

The code that you will use is

```
var mock = (IMatOps)new MatOpsMock

{

    MockAdd = (a, b) => a+b,

    MockDivision = (a,b)=> a/b

};
```

The code that is generated is

```
    public partial class MatOpsMock : global::MatOps.IMatOps

    {

        /// <summary>

        /// Set this to true, if you want members that don't have a mock
implementation

        /// to return a default value instead of throwing an exception.

        /// </summary>

        public bool ReturnDefaultIfNotMocked { get; set; }



        private System.Collections.Generic.List<HistoryEntry> historyEntries =
new System.Collections.Generic.List<HistoryEntry>();

        public System.Collections.ObjectModel.ReadOnlyCollection<HistoryEntry>
HistoryEntries

        {

            get

            {

                return historyEntries.AsReadOnly();

            }

        }







        /// <summary>

        /// Implemented for type global::MatOps.IMatOps (Public, same assembly:
False)

        /// </summary>

        public Func<int,int,int>? MockAdd { get; set; }
```

```
        public int Add(int a, int b)

        {

            historyEntries.Add(new HistoryEntry("Add", new [] { $"{a}", $"{b}"
}));



            if (MockAdd == null)

            {

                if (ReturnDefaultIfNotMocked)

                {

                    return default(int);

                }

                else

                {

                    throw new NotImplementedException("Method 'MockAdd' was
called, but no mock implementation was provided");

                }

            }



            return MockAdd(a, b);

        }



        /// <summary>

        /// Implemented for type global::MatOps.IMatOps (Public, same assembly:
False)

        /// </summary>
```

```
        public Func<int,int,int>? MockDivision { get; set; }

        public int Division(int a, int b)

        {

            historyEntries.Add(new HistoryEntry("Division", new [] { $"{a}", $"
{b}" }));



            if (MockDivision == null)

            {

                if (ReturnDefaultIfNotMocked)

                {

                    return default(int);

                }

                else

                {

                    throw new NotImplementedException("Method 'MockDivision'
was called, but no mock implementation was provided");

                }

            }



            return MockDivision(a, b);

        }

    }
```

Example Code: https://github.com/ignatandrei/RSCG_Examples/tree/main/DynamicMocking (https://github.com/ignatandrei/RSCG_Examples/tree/main/DynamicMocking)

All Generators: https://github.com/ignatandrei/RSCG_Examples/ (https://github.com/ignatandrei/RSCG_Examples/)

RSCG number 10 : Method decorator

Nuget :
https://www.nuget.org/packages/AOPMethodsCommon/
https://www.nuget.org/packages/AOPMethodsGenerator/

link : http://msprogrammer.serviciipeweb.ro/category/roslyn/

author :Andrei Ignat

What can do

This will generate code to decorate methods with anything you want (stopwatch, logging , authorization...)

The code that you start with is

```
    [AutoMethods(template =TemplateMethod.CustomTemplateFile,MethodPrefix
="prv" ,CustomTemplateFileName ="MethodDecorator.txt")]

    public partial class Person

    {

          public string FirstName{ get; set; }

          public string LastName { get; set; }



          private string prvFullName()

          {

                return FirstName + " " + LastName;

          }

    }
```

The code that you will use is

```
var p = new Person();
p.FirstName = "Andrei";
p.LastName = "Ignat";
Console.WriteLine(p.FullName());
```

The code that is generated is

```
[GeneratedCode("AOPMethods", "2021.2.22.1125")]
[CompilerGenerated]
public partial class Person{

     public  string FullName (

     [CallerMemberName] string memberName = "",
     [CallerFilePath] string sourceFilePath = "",
     [CallerLineNumber] int sourceLineNumber = 0){
          var sw=Stopwatch.StartNew();
          try{
               Console.WriteLine("--prvFullName start ");
               Console.WriteLine("called from class :"+memberName );
               Console.WriteLine("called from file :"+sourceFilePath );
               Console.WriteLine("called from line :"+sourceLineNumber );
                    prvFullName();
          }
```

```
            catch(Exception ex){

                Console.WriteLine("error in prvFullName:" + ex.Message);

                throw;

            }

            finally{

                Console.WriteLine($"--------prvFullName end in
{sw.Elapsed.TotalMilliseconds}");

            }




        }//end FullName




    }
```

Example Code: https://github.com/ignatandrei/RSCG_Examples/tree/main/MethodDecorator (https://github.com/ignatandrei/RSCG_Examples/tree/main/MethodDecorator)

All Generators: https://github.com/ignatandrei/RSCG_Examples/ (https://github.com/ignatandrei/RSCG_Examples/)

RSCG number 11 : PartiallyApplied

Nuget :
https://www.nuget.org/packages/PartiallyApplied/

link : https://github.com/JasonBock/PartiallyApplied

author :Andrei Ignat

What can do

This will generate curry for your functions

The code that you start with is

```
public class Accounting

{

    public static float Discount( float discount, float price)

    {

        var val= price * (1- discount);

        return val;

    }

}
```

The code that you will use is

```
var disc10Percent = Partially.Apply(Accounting.Discount, 1/10f);

Console.WriteLine(disc10Percent(disc10Percent(100)));
```

The code that is generated is

```
   public static partial class Partially

   {

           public static Func<float, float> Apply(Func<float, float, float>
method, float discount) =>

                   new((price) => method(discount, price));

   }
```

Example Code: https://github.com/ignatandrei/RSCG_Examples/tree/main/PartiallyFunction (https://github.com/ignatandrei/RSCG_Examples/tree/main/PartiallyFunction)

All Generators: https://github.com/ignatandrei/RSCG_Examples/ (https://github.com/ignatandrei/RSCG_Examples/)

RSCG number 12 : IFormattable

Nuget :
https://www.nuget.org/packages/AOPMethodsCommon/
https://www.nuget.org/packages/AOPMethodsGenerator/

link : http://msprogrammer.serviciipeweb.ro/category/roslyn/

author :Andrei Ignat

What can do

This will generate code to add IFormattable to any class, based on the properties of the class

The code that you start with is

```
    [AutoMethods(CustomTemplateFileName = "CreateFormattable.txt", template =
TemplateMethod.CustomTemplateFile)]

    partial class Department

    {

        public int ID { get; set; }

        public string Name { get; set; }



    }

    [AutoMethods(CustomTemplateFileName = "CreateFormattable.txt", template =
TemplateMethod.CustomTemplateFile)]

    partial class Employee

    {

        public int ID { get; set; }

        public string Name { get; set; }



        public Department dep { get; set; }



    }
```

The code that you will use is

```
    var e = new Employee();

    e.ID = 1;

    e.Name = "Andrei";

    e.dep = new Department();

    e.dep.Name = "IT";



    Console.WriteLine(e.ToString("for employee with id = {id} the name is
{name} and department is {dep?.Name}", null));



    e.dep = null;



    Console.WriteLine(e.ToString("for employee with id = {id} the name is
{name} and department is {dep?.Name}", null));
```

The code that is generated is

```
    [GeneratedCode("AOPMethods", "2021.2.27.640")]

    [DebuggerDisplay(" ID = {ID} Name = {Name} dep = {dep}")]

    partial class Employee: IFormattable{

        public object ValueProperty(string val){

            val = val.Replace("?","");


if(string.Compare("ID",val,StringComparison.CurrentCultureIgnoreCase)==0) {

                    return this.ID;
```

```
                }

if(string.Compare("Name",val,StringComparison.CurrentCultureIgnoreCase)==0) {
                    return this.Name;
                }

if(string.Compare("dep",val,StringComparison.CurrentCultureIgnoreCase)==0) {
                    return this.dep;
                }
            throw new ArgumentException("cannot find "+ val);
        }


        //adapted from https://haacked.com/archive/2009/01/14/named-formats-
redux.aspx/
        private object Eval(string expression,IFormatProvider formatProvider)
        {
            if (expression.Contains("."))
            {
                var splut = expression.Split(".");
                bool canBeNull=splut[0].Contains("?");
                dynamic d = ValueProperty(splut[0]);
                if(canBeNull && d == null)
                    return null;
                for(var i=1; i<splut.Length;i++){
                    canBeNull=splut[i].Contains("?");
                    d=d.ToString("{"+splut[i]+"}",formatProvider);
```

```
                if(canBeNull && d == null)
                    return null;
            }
            return d;

        }

        return ValueProperty(expression);

    }




    public string ToString(string format, IFormatProvider formatProvider)
    {
        if (format == null)
            throw new ArgumentNullException("format");

        List<object> values = new List<object>();
        string rewrittenFormat = Regex.Replace(format,
            delegate (Match m)
            {
                Group startGroup = m.Groups["start"];
                Group propertyGroup = m.Groups["property"];
```

```
                Group formatGroup = m.Groups["format"];
                Group endGroup = m.Groups["end"];

                values.Add((propertyGroup.Value == "0")
            ? this
            : Eval(propertyGroup.Value, formatProvider));

                int openings = startGroup.Captures.Count;
                int closings = endGroup.Captures.Count;

                return openings > closings || openings % 2 == 0
                ? m.Value
                : new string('{', openings) + (values.Count - 1)
                + formatGroup.Value
                + new string('}', closings);
                },
                RegexOptions.Compiled
                | RegexOptions.CultureInvariant
                | RegexOptions.IgnoreCase);

            return string.Format(formatProvider, rewrittenFormat,
values.ToArray());
        }
```

```
    }
```

Example Code: https://github.com/ignatandrei/RSCG_Examples/tree/main/IFormattable (https://github.com/ignatandrei/RSCG_Examples/tree/main/IFormattable)

All Generators: https://github.com/ignatandrei/RSCG_Examples/ (https://github.com/ignatandrei/RSCG_Examples/)

RSCG number 13 : AutoInterface

Nuget :
https://www.nuget.org/packages/BeaKona.AutoInterfaceGenerator

link : https://github.com/beakona/AutoInterface

author :beakona

What can do

Implement the Design Pattern Decorator. Based on template - you can modify the source code generated

The code that you start with is

```
public interface ICoffee

{

    public int Price { get; }

    public string Description { get; }

}



public class SimpleCoffee : ICoffee

{

    public SimpleCoffee()

    {

        Price = 3;

        Description = "Simple Coffee";

    }

    public int Price { get; set; }

    public string Description { get; set; }
```

```
    public partial class MilkDecorator : ICoffee

    {

        [BeaKona.AutoInterface(TemplateLanguage = "scriban", TemplateBody =
SimpleCoffee.TemplateCoffeeDecorator)]

        private readonly ICoffee coffee;



        public int DecoratorPrice { get; set; } = 1;

        public MilkDecorator(ICoffee coffee)

        {

            this.coffee = coffee;

        }









    }



    public partial class ChocoDecorator : ICoffee

    {

        [BeaKona.AutoInterface(TemplateLanguage = "scriban", TemplateBody =
SimpleCoffee.TemplateCoffeeDecorator)]

        private readonly ICoffee coffee;



        public int DecoratorPrice { get; set; } = 2;

        public ChocoDecorator(ICoffee coffee)
```

```
        {
            this.coffee = coffee;
        }



    }
```

The code that you will use is

```
    SimpleCoffee s = new SimpleCoffee();
    Console.WriteLine(s.Description +" with Price "+ s.Price);
    ICoffee withMilk = new MilkDecorator(s);
    Console.WriteLine(withMilk.Description} +" with Price "+ withMilk.Price);
    ICoffee withMilkAndChoco = new ChocoDecorator(withMilk);
    Console.WriteLine(withMilkAndChoco.Description +" with Price "+
withMilkAndChoco.Price);
```

The code that is generated is

```
    partial class MilkDecorator
    {
        int ICoffee.Price
        {
            get
            {
```

```
                return ((ICoffee)this.coffee).Price + DecoratorPrice;


            }
        }

        string ICoffee.Description
        {
            get
            {

                var name = this.GetType().Name.Replace("Decorator","");
                return ((ICoffee)this.coffee).Description + " with " +
name;


            }
        }
    }
```

Example Code: https://github.com/ignatandrei/RSCG_Examples/tree/main/DP_Decorator (https://github.com/ignatandrei/RSCG_Examples/tree/main/DP_Decorator)

All Generators: https://github.com/ignatandrei/RSCG_Examples/ (https://github.com/ignatandrei/RSCG_Examples/)

RSCG number 14 : Property Expression Generator

Nuget :
https://www.nuget.org/packages/AOPMethodsCommon/
https://www.nuget.org/packages/AOPMethodsGenerator/

link : http://msprogrammer.serviciipeweb.ro/category/roslyn/

author :Andrei Ignat

What can do

This will generate code to add function to be used with Entity Framework to search for any property of a class

The code that you start with is

```
    [AutoMethods(template = TemplateMethod.CustomTemplateFile,
CustomTemplateFileName = "CreateMetadata.txt")]

    public partial class Person

    {

        public int ID { get; set; }

        public string FirstName { get; set; }

        public string LastName { get; set; }

        public DateTime? DateOfBirth {get;set;}

    }
```

The code that you will use is

```
    var queryCnt = Metadata_Person.expr_FirstName_Contains("9");

    var pers= await cnt.Person.Where(queryCnt).ToArrayAsync();

    Console.WriteLine(pers.Length);
```

```
queryCnt = Metadata_Person.expr_LastName_NullOrWhite();
pers = await cnt.Person.Where(queryCnt).ToArrayAsync();
Console.WriteLine(pers.Length);



var queryID = Metadata_Person.expr_ID_Equal(7);
var pId = await cnt.Person.FirstOrDefaultAsync(queryID);
Console.WriteLine(pId.FirstName);



queryID = Metadata_Person.expr_ID_Contains(7,9);
pers = await cnt.Person.Where(queryID).ToArrayAsync();
Console.WriteLine(pers.Length);



var nullBirthDateQuery = Metadata_Person.expr_DateOfBirth_Null();
var birthNull = await cnt.Person.Where(nullBirthDateQuery).ToArrayAsync();
Console.WriteLine(birthNull.Length);

var query = Metadata_Person.FindEx("ID", SearchCriteria.Equal, 99);
pers = await cnt.Person.Where(query).ToArrayAsync();
Console.WriteLine(pers.Length);
```

```
    query = Metadata_Person.FindEx("DateOfBirth", SearchCriteria.FindNull);
    pers = await cnt.Person.Where(query).ToArrayAsync();
    Console.WriteLine(pers.Length);
```

The code that is generated is

```
    [CompilerGenerated]
    public partial class Metadata_Person{




        //public const string prop_ID = "ID";
        //public static readonly Func<Person,int> func_ID = (it=>it.ID);
        //public static readonly Expression<Func<Person,int>> expr_ID =
(it=>it.ID);
        public static Expression<Func<Person,bool>> expr_ID_Equal(int value)=>
(it=>it.ID == value);
        public static Expression<Func<Person,bool>> expr_ID_Diff(int value)=>
(it=>it.ID != value);
        public static Expression<Func<Person,bool>> expr_ID_Contains(params
int[] value)=> (it=> value.Contains(it.ID) );


        //int
```

```
        public static Expression<Func<Person,bool>> expr_ID_Greater(int
value)=> (it=>it.ID > value);

        public static Expression<Func<Person,bool>> expr_ID_GreaterOrEqual(int
value)=> (it=>it.ID >= value);

        public static Expression<Func<Person,bool>> expr_ID_Less(int value)=>
(it=>it.ID < value);

        public static Expression<Func<Person,bool>> expr_ID_LessOrEqual(int
value)=> (it=>it.ID <= value);









        //public const string prop_FirstName = "FirstName";

        //public static readonly Func<Person,string> func_FirstName =
(it=>it.FirstName);

        //public static readonly Expression<Func<Person,string>> expr_FirstName
= (it=>it.FirstName);

        public static Expression<Func<Person,bool>> expr_FirstName_Equal(string
value)=> (it=>it.FirstName == value);

        public static Expression<Func<Person,bool>> expr_FirstName_Diff(string
value)=> (it=>it.FirstName != value);

        public static Expression<Func<Person,bool>>
expr_FirstName_Contains(params string[] value)=> (it=>
value.Contains(it.FirstName) );
```

```
        //string




        public static Expression<Func<Person,bool>> expr_FirstName_Null()=>
(it=>it.FirstName == null);









        public static Expression<Func<Person,bool>>
expr_FirstName_NullOrWhite()=> (it=>string.IsNullOrWhiteSpace(it.FirstName));



        public static Expression<Func<Person,bool>> expr_FirstName_Ends(string
value)=> (it=>it.FirstName.StartsWith (value));

        public static Expression<Func<Person,bool>>
expr_FirstName_Starts(string value)=> (it=>it.FirstName.EndsWith(value));

        public static Expression<Func<Person,bool>>
expr_FirstName_Contains(string value)=> (it=>it.FirstName.Contains(value));









        //public const string prop_LastName = "LastName";

        //public static readonly Func<Person,string> func_LastName =
```

```
(it=>it.LastName);

        //public static readonly Expression<Func<Person,string>> expr_LastName
= (it=>it.LastName);

        public static Expression<Func<Person,bool>> expr_LastName_Equal(string
value)=> (it=>it.LastName == value);

        public static Expression<Func<Person,bool>> expr_LastName_Diff(string
value)=> (it=>it.LastName != value);

        public static Expression<Func<Person,bool>>
expr_LastName_Contains(params string[] value)=> (it=>
value.Contains(it.LastName) );



        //string









        public static Expression<Func<Person,bool>> expr_LastName_Null()=>
(it=>it.LastName == null);









        public static Expression<Func<Person,bool>>
expr_LastName_NullOrWhite()=> (it=>string.IsNullOrWhiteSpace(it.LastName));



        public static Expression<Func<Person,bool>> expr_LastName_Ends(string
value)=> (it=>it.LastName.StartsWith (value));

        public static Expression<Func<Person,bool>> expr_LastName_Starts(string
value)=> (it=>it.LastName.EndsWith(value));

        public static Expression<Func<Person,bool>>
```

```
expr_LastName_Contains(string value)=> (it=>it.LastName.Contains(value));









        //public const string prop_DateOfBirth = "DateOfBirth";

        //public static readonly Func<Person,System.DateTime?> func_DateOfBirth
= (it=>it.DateOfBirth);

        //public static readonly Expression<Func<Person,System.DateTime?>>
expr_DateOfBirth = (it=>it.DateOfBirth);

        public static Expression<Func<Person,bool>>
expr_DateOfBirth_Equal(System.DateTime? value)=> (it=>it.DateOfBirth == value);

        public static Expression<Func<Person,bool>>
expr_DateOfBirth_Diff(System.DateTime? value)=> (it=>it.DateOfBirth != value);

        public static Expression<Func<Person,bool>>
expr_DateOfBirth_Contains(params System.DateTime?[] value)=> (it=>
value.Contains(it.DateOfBirth) );



        //System.DateTime?









        public static Expression<Func<Person,bool>> expr_DateOfBirth_Null()=>
(it=>it.DateOfBirth == null);
```

```
        public static Expression<Func<Person,bool>>
expr_DateOfBirth_Greater(System.DateTime? value)=> (it=>it.DateOfBirth >
value);

        public static Expression<Func<Person,bool>>
expr_DateOfBirth_GreaterOrEqual(System.DateTime? value)=> (it=>it.DateOfBirth
>= value);

        public static Expression<Func<Person,bool>>
expr_DateOfBirth_Less(System.DateTime? value)=> (it=>it.DateOfBirth < value);

        public static Expression<Func<Person,bool>>
expr_DateOfBirth_LessOrEqual(System.DateTime? value)=> (it=>it.DateOfBirth <=
value);









        public static Expression<Func<Person,bool>> FindEx(string nameProp,
SearchCriteria search, object value = null)

        {









if(string.Compare("ID",nameProp,StringComparison.CurrentCultureIgnoreCase) ==
0)

            switch(search){

                case SearchCriteria.None:

                    return null;
```

```
                case SearchCriteria.Equal:
                    var orig= (int) value;
                    return expr_ID_Equal(orig);
                default:
                    throw new ArgumentException("cannot find for ID case
"+search);
            }




if(string.Compare("FirstName",nameProp,StringComparison.CurrentCultureIgnoreCas
e) == 0)
            switch(search){
                case SearchCriteria.None:
                    return null;

                case SearchCriteria.FindNull:
                    return expr_FirstName_Null();

                case SearchCriteria.Equal:
                    var orig= (string) value;
                    return expr_FirstName_Equal(orig);
                default:
```

```
                    throw new ArgumentException("cannot find for FirstName case
"+search);

            }









if(string.Compare("LastName",nameProp,StringComparison.CurrentCultureIgnoreCase
) == 0)

            switch(search){

                case SearchCriteria.None:

                    return null;



                case SearchCriteria.FindNull:

                    return expr_LastName_Null();



                case SearchCriteria.Equal:

                    var orig= (string) value;

                    return expr_LastName_Equal(orig);

                default:

                    throw new ArgumentException("cannot find for LastName case
"+search);

            }
```

```
if(string.Compare("DateOfBirth",nameProp,StringComparison.CurrentCultureIgnoreCase) == 0)

            switch(search){

                case SearchCriteria.None:

                    return null;



                case SearchCriteria.FindNull:

                    return expr_DateOfBirth_Null();



                case SearchCriteria.Equal:

                    var orig= (System.DateTime?) value;

                    return expr_DateOfBirth_Equal(orig);

                default:

                    throw new ArgumentException("cannot find for DateOfBirth case  "+search);

            }




            throw new ArgumentException("cannot find property  "+nameProp);



        }


    }
```

Examples of useful Roslyn Source Code Generator

Example Code:
https://github.com/ignatandrei/RSCG_Examples/tree/main/PropertyExpressionGenerator
(https://github.com/ignatandrei/RSCG_Examples/tree/main/PropertyExpressionGenerator)

All Generators: https://github.com/ignatandrei/RSCG_Examples/
(https://github.com/ignatandrei/RSCG_Examples/)

Examples of useful Roslyn Source Code Generator

RSCG - worth mention

There are more RSCG that you could see - here is a list that you may want to look at:

1. AutoEmbed https://github.com/chsienki/AutoEmbed
2. Cloneable https://github.com/mostmand/Cloneable
3. fonderie https://github.com/jeromelaban/fonderie
4. Generators.Blazor https://github.com/excubo-ag/Generators.Blazor
5. Generators.Grouping https://github.com/excubo-ag/Generators.Grouping
6. JsonMergePatch https://github.com/ladeak/JsonMergePatch
7. MemoizeSourceGenerator https://github.com/Zoxive/MemoizeSourceGenerator
8. MiniRazor https://github.com/Tyrrrz/MiniRazor/
9. MockGen https://github.com/thomas-girotto/MockGen
10. ProxyGen https://github.com/Sholtee/ProxyGen
11. Rocks https://github.com/JasonBock/Rocks
12. RoslynWeave https://github.com/Jishun/RoslynWeave
13. SmallSharp https://github.com/devlooped/SmallSharp
14. StaticProxyGenerator https://github.com/robertturner/StaticProxyGenerator
15. ValueChangedGenerator https://github.com/ufcpp/ValueChangedGenerator
16. Web-Anchor https://github.com/mattiasnordqvist/Web-Anchor
17. WrapperValueObject https://github.com/martinothamar/WrapperValueObject