

Introduction

Overview

MavFiFoundation.SourceGenerators

An extendable source generator that builds on [Roslyn's](#) support for [incremental generators](#).

Basic functionality

Source generation can be triggered by specifying configuration information using:

- Type Attributes
- Yaml Files
- Json Files
- XML Files

The following template languages can be used to specify generated source code:

- [Scriban](#)
- [Liquid](#) (via Scriban's [Liquid support](#))

Basic usage

Adding the following trigger file would generate an EF Core `DbContext` for all classes in the project that include an attribute named `DbEntity`.

Trigger file (`ExampleContext.CodeGen.yml`)

```
srcLocatorType: MFFAttributeTypeLocator
srcLocatorInfo: Example.Data.DbEntityAttribute
genOutputInfos:
  - fileNameBuilderInfo: 'ExampleDbContext.g.cs'
    sourceBuilderType: MFFScribanBuilder
    sourceBuilderInfo: |-
      #nullable enable

      using Microsoft.EntityFrameworkCore;

      namespace Example.Data

      public partial class ExampleContext : DbContext
      {
```

```

        public ExampleContext(DbContextOptions<ExampleContext> options)
: base(options)
    {
    }

    {{- for srcType in srcTypes }}
        public DbSet<{{ srcType.Name }}> {{ srcType.Name }}s { get; set; }

    {{- end }}
    }

```

Generated file (ExampleDbContext.g.cs)

```

#nullable enable

using Microsoft.EntityFrameworkCore;

namespace Example.Data

public partial class ExampleContext : DbContext
{
    public ExampleContext(DbContextOptions<ExampleContext> options)
: base(options)
    {

    }

    public DbSet<Example> Examples { get; set; }

    public DbSet<AdditionalExample> AdditionalExamples { get; set; }

}

```

Main Source Generator Components

Source generation involves three main components: [Generator Triggers](#), [Type Locators](#), and [Builders](#). Source generation begins with and is configured using generator triggers. Once source generation has been triggered by a generator trigger, control is passed to type locators that are used to locate the types that will be passed to builders. The builders will then create the source code.

Generator Triggers

Generator triggers are the starting point for source generation and are where the configuration is defined for code to be generated. Regardless of the specific generator trigger that is used, the supplied configuration information is used to create a [MFFGeneratorInfoModel](#). How this information is specified is depending on the specific generator trigger that is being used. The following generator triggers are included:

- [MFFAttributeGeneratorTrigger](#) - Use an attribute on a type to trigger source generation.
- [MFFYamlGeneratorTrigger](#) - Use a yaml file with a CodeGen.yml or CodeGen.yaml extension to trigger source generation.
- [MFFJsonGeneratorTrigger](#) - Use a json file with a CodeGen.json extension to trigger source generation.
- [MFFXmlGeneratorTrigger](#) - Use a xml file with a CodeGen.xml extension to trigger source generation.

Type Locators

Type locators are used to locate the types that will be used to generate source from. The located types are passed to a builder that can use the various parts of existing types to create source code for new types. The type locator to use is specified in the [MFFGeneratorInfoModel.SrcLocatorType](#) property. The configuration information used by the type locator is specified in the [MFFGeneratorInfoModel.SrcLocatorInfo](#) property and varies depending on the specific type locator being used. The following type locators are included:

- [MFFAttributeTypeLocator](#) - Use an attribute on a type to locate types.
- [MFFDynamicLinqTypeLocator](#) - Use a [Dynamic LINQ](#) query to locate types.
- [MFFIncludedTypeLocator](#) - Use the type that was used by the generator trigger.

Builders

Builder are used to generate the source. Builders are also used for generating the filename for the generated source. Configuration information for builders is supplied using [MFFBuilderModel](#). The builder to use for source generation is specified in the [MFFBuilderModel.SourceBuilderType](#) property. The builder to use for filename generation is specified in the [MFFBuilderModel.FileNameBuilderType](#) property. The following builders are included:

- [MFFScribanBuilder](#) - Use a [Scriban](#) template to build source.
- [MFFLiquidBuilder](#) - Use a [Liquid](#) template to build source.

Getting Started

Coming soon!

MavFiFoundation.SourceGenerators

Contains the core functionality for creating source generator implementations and source generator plugins.

* See [MavFiFoundation.SourceGenerators.MFFSourceGenerator](#) for a ready-to-use concrete source generator implementation.

MavFiFoundationSourceGenerators.

MFFSourceGenerator

Contains a ready-to-use concrete source generator implementation configured to use all of the plugins available in the [MavFiFoundation.SourceGenerator](#) project.

Basic functionality

Source generation can be triggered by specifying configuration information using:

- Type Attributes
- Yaml Files
- Json Files
- XML Files

The following template languages can be used to specify generated source code:

- [Scriban](#)
- [Liquid](#) (via Scriban's [Liquid support](#))

Basic usage

Adding the following trigger file would generate an EF Core `DbContext` for all classes in the project that include an attribute named `DbEntity`.

Trigger file (`ExampleContext.CodeGen.yml`)

```
srcLocatorType: MFFAttributeTypeLocator
srcLocatorInfo: Example.Data.DbEntityAttribute
genOutputInfos:
  - fileNameBuilderInfo: 'ExampleDbContext.g.cs'
    sourceBuilderType: MFFScribanBuilder
    sourceBuilderInfo: |-
      #nullable enable

      using Microsoft.EntityFrameworkCore;

      namespace Example.Data

      public partial class ExampleContext : DbContext
      {
        public ExampleContext(DbContextOptions<ExampleContext> options)
        : base(options)
        {
```

```

    }

    {{- for srcType in srcTypes }}
        public DbSet<{{ srcType.Name }}> {{ srcType.Name }}s { get; set; }

    {{- end }}
}

```

Generated file (ExampleDbContext.g.cs)

```

#nullable enable

using Microsoft.EntityFrameworkCore;

namespace Example.Data

public partial class ExampleContext : DbContext
{
    public ExampleContext(DbContextOptions<ExampleContext> options) : base(options)
    {

    }

    public DbSet<Example> Examples { get; set; }

    public DbSet<AdditionalExample> AdditionalExamples { get; set; }

}

```

MavFiFoundation.SourceGenerators.Shared

Contains classes that may be useful to the target project, automated tests, and other tools.

MavFiFoundation.SourceGenerators.Extras

Contains classes that may be useful to the target project, automated tests, and other tools, but rely on a concrete MavFiFoundation.SourceGenerator implementation to build.

MavFiFoundation.SourceGenerators.Testing

Contains classes for testing source generators that may be used by any testing framework.

MavFiFoundation.SourceGenerators.Testing.XUnit

Contains classes for testing source generators that are specific to the xunit testing framework.

MFFAttributeGeneratorTrigger

Uses an attribute ([MFFGenerateSourceAttribute](#) by default) added to a type to trigger source generation. Configuration information is specified using the constructor for the attribute.

The [API](#) documentation contains [examples](#).

MFFYamlGeneratorTrigger

Uses a yaml file with a CodeGen.yaml or CodeGen.yml extension to trigger source generation. The yaml file includes a serialized instance of [MFFGeneratorInfoModel](#) that contains the configuration information needed by the source generator.

The [API](#) documentation contains [examples](#).

MFFJsonGeneratorTrigger

Uses a json file with a CodeGen.json extension to trigger source generation. The json file includes a serialized instance of [MFFGeneratorInfoModel](#) that contains the configuration information needed by the source generator.

The [API](#) documentation contains [examples](#).

MFFXmlGeneratorTrigger

Uses an xml file with a CodeGen.xml extension to trigger source generation. The xml file includes a serialized instance of [MFFGeneratorInfoModel](#) that contains the configuration information needed by the source generator.

The [API](#) documentation contains [examples](#).

MFFAttributeTypeLocator

Used to locate types by an attribute added to the type. The attribute to use is specified as a string in the [MFFGeneratorInfoModel.SrcLocatorType](#) property provided to the generator trigger.

For additional information, see the [API](#) documentation.

MFFDynamicLinqTypeLocator

Used to locate types using a [Dynamic LINQ](#) query. The where query to use can be specified as a string in the [MFFGeneratorInfoModel.SrcLocatorType](#) property provided to the generator trigger or it may be provided in the [LinqWhere](#) property of a serialized instance of [MFFDynamicLinqTypeLocatorInfo](#).

For additional information, see the [API](#) documentation.

- [MFFDynamicLinqTypeLocator](#).

MFFIncludedTypeLocator

Used when the type is included in the [MFFGeneratorInfoModel.SrcLocatorType](#) property by the generator trigger.

For additional information, see the [API](#) documentation.

MFFScribanBuilder

Uses a [Scriban](#) template to build the required output.

For additional information, see the [API](#) documentation.

MFFLiquidBuilder

Uses a [Liquid](#) template to build the required output. Liquid template support is provided by Scriban's built in [Liquid Support](#)].

For additional information, see the [API](#) documentation.

Namespace MavFiFoundation.Source Generators

Classes

[CreateGeneratorConstants](#)

[EquatableArray](#)

Extensions for [EquatableArray<T>](#).

[INamedTypeSymbolExtensions](#)

[MFFCreateGeneratorConstantsAttribute](#)

[MFFEmbeddedResourceAttribute](#)

[MFFGenerateSourceAttribute](#)

[MFFGenerator](#)

[MFFGeneratorAnalyzer](#)

[MFFGeneratorAnalyzerBase](#)

[MFFGeneratorBase](#)

[MFFGeneratorConstants](#)

[MFFGeneratorConstants.Generator](#)

[MFFGeneratorHelper](#)

[MFFGeneratorHelperBase](#)


[MFFGeneratorPluginBase](#)

The base class that all [IMFFGeneratorPlugin](#) implementations SHOULD implement.

[MFFGeneratorPluginsProvider](#)

Structs

[EquatableArray<T>](#)

An immutable, equatable array. This is equivalent to [ImmutableArray<T>](#)  but with value equality support.

Interfaces

[IMFFGeneratorHelper](#)

Defines helper methods for [MFFGeneratorBase](#) instances.

[IMFFGeneratorPlugin](#)

Defines the interface to be used by all generator plugins.

[IMFFGeneratorPluginsProvider](#)