Introduction

Getting Started

Field Feature

Auto properties only allow for directly setting or getting the backing field, giving some control only by placing access modifiers on the accessors.

Sometimes there is a need to have additional control over what happens in one or both accessors, but this confronts users with **the overhead of declaring a backing field**.

The backing field name must then be kept in sync with the property, and the backing field is scoped to the entire class which can result in accidental bypassing of the accessors from within the class.

This is the one of implementation:

```
[FieldFeature]
public partial class LegacyModel
{
    private int _number;
    public int Number
    {
        get => _number;
        set => _number = value;
    }

    private int[] _numbers;
    public int[] Numbers
    {
        get => _numbers;
        set => _numbers = value;
    }
}
```

However, we hope to **avoid manually declaring backing fields** for older projects.

Use **FieldFeature** to enable the generator to analyze the code of a class or struct, and use **FieldBackend** to hide the backing fields:

```
[FieldFeature]
public partial class Model
{
    [FieldBackend]
    public int Number
    {
```

```
        get => GetValue();
        set => SetValue(value);
    }

    [FieldBackend]
    public int[] Numbers
    {
        get => GetValue();
        set => SetValue(value);
    }
}
```

.NET 10 (C# 14) added the **field** keyword to handle this situation.

```
public class Model
{
    public int Number
    {
        get => field;
        set => field = value;
    }

    public int[] Numbers
    {
        get => field;
        set => field = value;
    }
}
```