KAJ VAS ROSLYN ANALIZATORJI LAHKO NAUČIJO O .NETU?

Damir Arh, Razum d.o.o.

Microsoft MVP

O MENI

- Razum d.o.o.
- Microsoft MVP
- https://damirscorner.com
- @DamirArh@mas.to
- @DamirArh

AGENDA

- Primeri analizatorjev
    - Regularni izrazi (SYSLIB1045)
    - Strukturirano beleženje (CA2254)
    - Beleženje z delegati (CA1848)
- Nastavitve

REGULARNI IZRAZI

Demo

REGULARNI IZRAZI: KAJ?

> *Use `GeneratedRegexAttribute` to generate the regular expression implementation at compile-time.*

- SYSLIB1045

REGULARNI IZRAZI: ZAKAJ?

Note

Where possible, use source-generated regular expressions instead of compiling regular expressions using the `RegexOptions.Compiled` option. Source generation can help your app start faster, run more quickly, and be more trimmable.

REGULARNI IZRAZI: UPORABA

```
Regex.IsMatch(input, pattern);
```

```
private static readonly Regex regex = new(pattern);

regex.IsMatch(input);
```

```
private static readonly Regex compiledRegex =
    new(pattern, RegexOptions.Compiled);

compiledRegex.IsMatch(input);
```

```
[GeneratedRegex(pattern)]
private static partial Regex SourceGeneratedRegex();

SourceGeneratedRegex().IsMatch(input);
```

REGULARNI IZRAZI: HITROST

Method	Mean	Error	StdDev
Regex.IsMatch	71.96 ns	0.301 ns	0.267 ns
new Regex()	68.44 ns	0.397 ns	0.331 ns
RegexOptions.Compiled	24.49 ns	0.070 ns	0.066 ns
GeneratedRegex	20.34 ns	0.048 ns	0.042 ns

STRUKTURIRANO BELEŽENJE

Demo

STRUKTURIRANO BELEŽENJE: KAJ?

> *The logging message template should not vary between calls*

- CA2254

STRUKTURIRANO BELEŽENJE: ZAKAJ?

Rule description

When performing logging, it's desirable to preserve the structure of the log (including placeholder names) along with the placeholder values. Preserving this information allows for better observability and search in log aggregation and monitoring software.

STRUKTURIRANO BELEŽENJE: KAKO?

How to fix violations

Update the message template to be a constant expression. If you're using values directly in the template, refactor the template to use named placeholders instead.

STRUKTURIRANO BELEŽENJE: PREJ

```
logger.LogInformation($"Hello, world from {name}!");
```

```
{
  "EventId": 0,
  "LogLevel": "Information",
  "Category": "DotNetAnalyzers.ConsoleLogger",
  "Message": "Hello, world from NTK!",
  "State": {
    "Message": "Hello, world from NTK!",
    "{OriginalFormat}": "Hello, world from NTK!"
  }
}
```

STRUKTURIRANO BELEŽENJE: POTEM

```
logger.LogInformation("Hello, world from {Name}!", name);
```

```
{
  "EventId": 0,
  "LogLevel": "Information",
  "Category": "DotNetAnalyzers.LoggingTests",
  "Message": "Hello, world from NTK!",
  "State": {
    "Message": "Hello, world from NTK!",
    "Name": "NTK",
    "{OriginalFormat}": "Hello, world from {Name}!"
  }
}
```

BELEŽENJE Z DELEGATI

Demo

BELEŽENJE Z DELEGATI: KAJ?

> *For improved performance, use the `LoggerMessage` delegates instead of calling `LoggerExtensions`*

- CA1848

BELEŽENJE Z DELEGATI: ZAKAJ?

High-performance logging in .NET

The LoggerMessage class exposes functionality to create cacheable delegates that require fewer object allocations and reduced computational overhead compared to logger extension methods, such as LogInformation and LogDebug. For high-performance logging scenarios, use the LoggerMessage pattern.

BELEŽENJE Z DELEGATI: KAKO?

Important

Instead of using the LoggerMessage class to create high-performance logs, you can use the LoggerMessage attribute in .NET 6 and later versions. The `LoggerMessageAttribute` provides source-generation logging support designed to deliver a highly usable and highly performant logging solution for modern .NET applications.

BELEŽENJE Z DELEGATI: PREJ

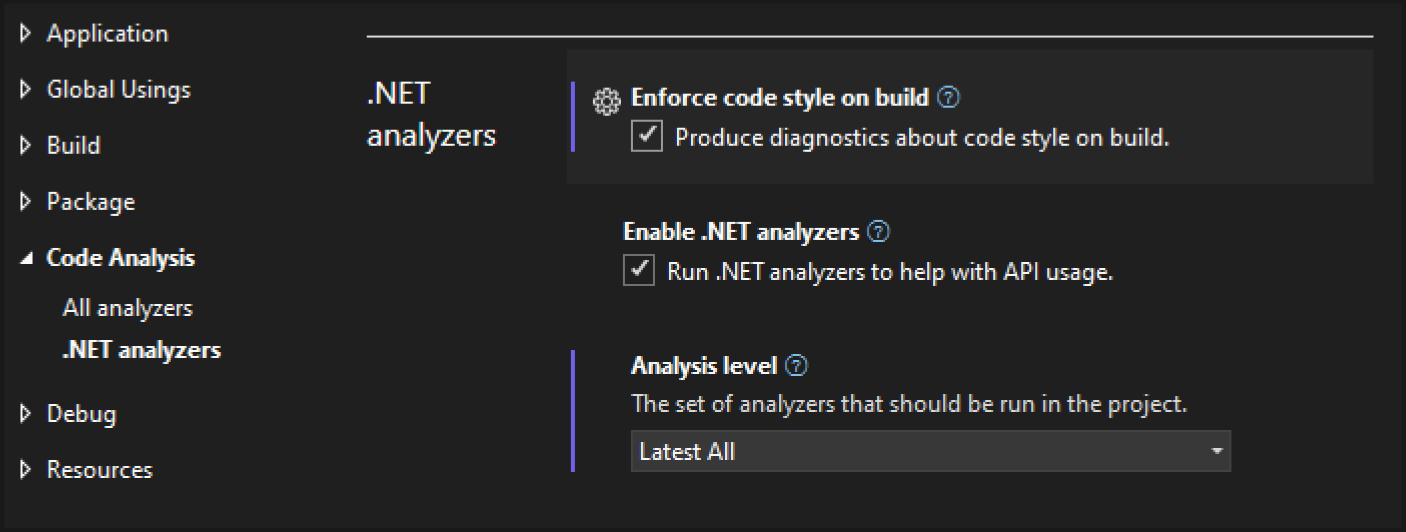
```
logger.LogInformation("Hello, world from {Name}!", name);
```

BELEŽENJE Z DELEGATI: POTEM

```
logger.HelloWorld(name);

public static partial class LoggerExtensions
{
    [LoggerMessage(
        EventId = 1,
        Level = LogLevel.Information,
        Message = "Hello, world from {Name}!"
    )]
    public static partial void HelloWorld(
        this ILogger logger,
        string name
    );
}
```

NASTAVITVE: NIVO ANALIZE



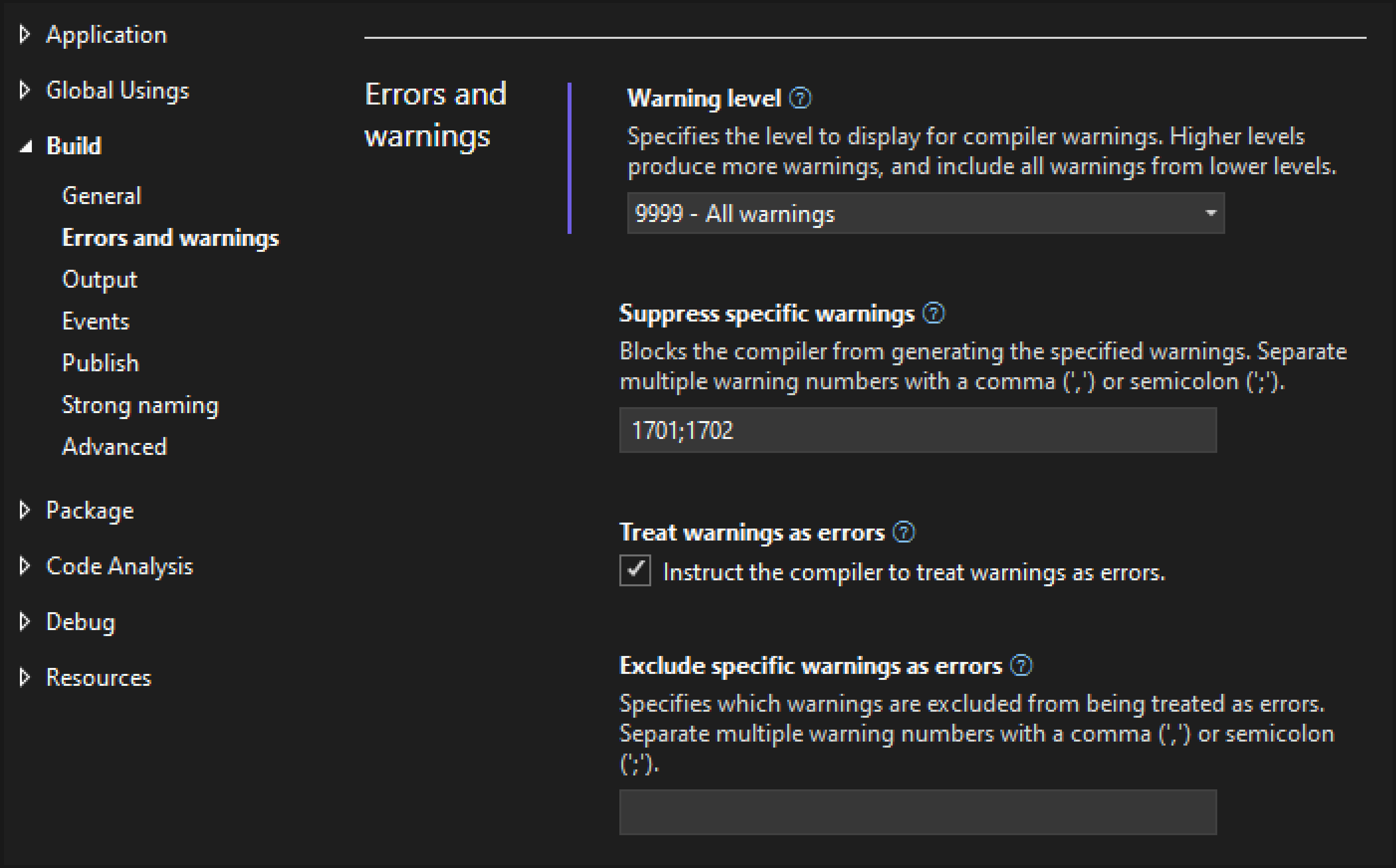
.NET analyzers

Enforce code style on build
[✓] Produce diagnostics about code style on build.

Enable .NET analyzers
[✓] Run .NET analyzers to help with API usage.

Analysis level
The set of analyzers that should be run in the project.
Latest All

NASTAVITVE: OPOZORILA IN NAPAKE



Errors and warnings

Warning level
Specifies the level to display for compiler warnings. Higher levels produce more warnings, and include all warnings from lower levels.
9999 - All warnings

Suppress specific warnings
Blocks the compiler from generating the specified warnings. Separate multiple warning numbers with a comma (',') or semicolon (';').
1701;1702

Treat warnings as errors
[✓] Instruct the compiler to treat warnings as errors.

Exclude specific warnings as errors
Specifies which warnings are excluded from being treated as errors. Separate multiple warning numbers with a comma (',') or semicolon (';').

NASTAVITVE: PROJEKTNA DATOTEKA

`*.csproj`:

```xml
<PropertyGroup>
  <AnalysisLevel>latest-all</AnalysisLevel>
  <EnforceCodeStyleInBuild>True</EnforceCodeStyleInBuild>
  <WarningLevel>9999</WarningLevel>
  <TreatWarningsAsErrors>True</TreatWarningsAsErrors>
</PropertyGroup>
```

NASTAVITVE: INDIVIDUALNI ANALIZATORJI

`.editorconfig`:

```
[*.cs]
# SYSLIB1045: Convert to 'GeneratedRegexAttribute'.
dotnet_diagnostic.SYSLIB1045.severity = none
```

PRIPOROČILA

- preberite dokumentacijo analizatorjev
- ne ignorirajte diagnoz
    - bodisi odpravite pomanjkljivost
    - bodisi izklopite analizo
- preprečite kopičenje opozoril

VIRI

- damirscorner.com/link/
  - AnalyzersGitHub
  - AnalyzersRegex
  - AnalyzersStructuredLog
  - AnalyzersHighPerfLog
  - AnalyzersConfig