

## LinksPlatform's Platform.RegularExpressions.Transformer Class Library

./Platform.RegularExpressions.Transformer/Context.cs

```
1  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
2
3  namespace Platform.RegularExpressions.Transformer
4  {
5      public class Context : IContext
6      {
7          public string Path { get; }
8
9          public Context(string path) => Path = path;
10     }
11 }
```

./Platform.RegularExpressions.Transformer/IContext.cs

```
1  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
2
3  namespace Platform.RegularExpressions.Transformer
4  {
5      public interface IContext
6      {
7          public string Path { get; }
8      }
9  }
```

./Platform.RegularExpressions.Transformer/ISubstitutionRule.cs

```
1  using System.Text.RegularExpressions;
2
3  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
4
5  namespace Platform.RegularExpressions.Transformer
6  {
7      public interface ISubstitutionRule
8      {
9          Regex MatchPattern { get; }
10         string SubstitutionPattern { get; }
11         Regex PathPattern { get; }
12         int MaximumRepeatCount { get; }
13     }
14 }
```

./Platform.RegularExpressions.Transformer/ITransformer.cs

```
1  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
2
3  namespace Platform.RegularExpressions.Transformer
4  {
5      public interface ITransformer
6      {
7          string Transform(string source, IContext context);
8      }
9  }
```

./Platform.RegularExpressions.Transformer/obj/Release/netstandard2.1/Platform.RegularExpressions.Transformer.AssemblyInfo.cs

```
1  //-----
2  // <auto-generated>
3  //     Generated by the MSBuild WriteCodeFragment class.
4  // </auto-generated>
5  //-----
6
7  using System;
8  using System.Reflection;
9
10 [assembly: System.Reflection.AssemblyConfigurationAttribute("Release")]
11 [assembly: System.Reflection.AssemblyCopyrightAttribute("Konstantin Diachenko")]
12 [assembly: System.Reflection.AssemblyDescriptionAttribute("LinksPlatform\'s
    ↪ Platform.RegularExpressions.Transformer Class Library")]
13 [assembly: System.Reflection.AssemblyFileVersionAttribute("0.0.2.0")]
14 [assembly: System.Reflection.AssemblyInformationalVersionAttribute("0.0.2")]
15 [assembly: System.Reflection.AssemblyTitleAttribute("Platform.RegularExpressions.Transformer")]
16 [assembly: System.Reflection.AssemblyVersionAttribute("0.0.2.0")]
```

./Platform.RegularExpressions.Transformer/RegexExtensions.cs

```
1  using System;
2  using System.Text.RegularExpressions;
3
4  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
5
6  namespace Platform.RegularExpressions.Transformer
```

```

7 {
8     public static class RegexExtensions
9     {
10         public static Regex OverrideOptions(this Regex regex, RegexOptions options, TimeSpan
            ↳ matchTimeout)
11         {
12             if (regex == null)
13             {
14                 return null;
15             }
16             return new Regex(regex.ToString(), options, matchTimeout);
17         }
18     }
19 }

```

./Platform.RegularExpressions.Transformer/SubstitutionRule.cs

```

1 using System;
2 using System.Text.RegularExpressions;
3
4 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
5
6 namespace Platform.RegularExpressions.Transformer
7 {
8     public class SubstitutionRule : ISubstitutionRule
9     {
10         public static readonly TimeSpan DefaultMatchTimeout = TimeSpan.FromMinutes(5);
11         public static readonly RegexOptions DefaultMatchPatternRegexOptions =
            ↳ RegexOptions.Compiled | RegexOptions.Multiline;
12         public static readonly RegexOptions DefaultPathPatternRegexOptions =
            ↳ RegexOptions.Compiled | RegexOptions.Singleline;
13
14         public Regex MatchPattern { get; set; }
15
16         public string SubstitutionPattern { get; set; }
17
18         public Regex PathPattern { get; set; }
19
20         public int MaximumRepeatCount { get; set; }
21
22         public SubstitutionRule(Regex matchPattern, string substitutionPattern, Regex
            ↳ pathPattern, int maximumRepeatCount, RegexOptions? matchPatternOptions,
            ↳ RegexOptions? pathPatternOptions, TimeSpan? matchTimeout)
23         {
24             MatchPattern = matchPattern;
25             SubstitutionPattern = substitutionPattern;
26             PathPattern = pathPattern;
27             MaximumRepeatCount = maximumRepeatCount;
28             OverrideMatchPatternOptions(matchPatternOptions ?? matchPattern.Options,
            ↳ matchTimeout ?? matchPattern.MatchTimeout);
29             OverridePathPatternOptions(pathPatternOptions ?? pathPattern.Options, matchTimeout
            ↳ ?? pathPattern.MatchTimeout);
30         }
31
32         public SubstitutionRule(Regex matchPattern, string substitutionPattern, Regex
            ↳ pathPattern, int maximumRepeatCount, bool useDefaultOptions) : this(matchPattern,
            ↳ substitutionPattern, pathPattern, maximumRepeatCount, useDefaultOptions ?
            ↳ DefaultMatchPatternRegexOptions : (RegexOptions?)null, useDefaultOptions ?
            ↳ DefaultPathPatternRegexOptions : (RegexOptions?)null, useDefaultOptions ?
            ↳ DefaultMatchTimeout : (TimeSpan?)null) { }
33
34         public SubstitutionRule(Regex matchPattern, string substitutionPattern, Regex
            ↳ pathPattern, int maximumRepeatCount) : this(matchPattern, substitutionPattern,
            ↳ pathPattern, maximumRepeatCount, true) { }
35
36         public SubstitutionRule(Regex matchPattern, string substitutionPattern, int
            ↳ maximumRepeatCount) : this(matchPattern, substitutionPattern, null,
            ↳ maximumRepeatCount) { }
37
38         public SubstitutionRule(Regex matchPattern, string substitutionPattern) :
            ↳ this(matchPattern, substitutionPattern, null, 0) { }
39
40         public static implicit operator SubstitutionRule(ValueTuple<Regex, string> tuple) => new
            ↳ SubstitutionRule(tuple.Item1, tuple.Item2);
41
42         public static implicit operator SubstitutionRule(ValueTuple<Regex, string, int> tuple)
            ↳ => new SubstitutionRule(tuple.Item1, tuple.Item2, tuple.Item3);
43
44         public static implicit operator SubstitutionRule(ValueTuple<Regex, string, Regex, int>
            ↳ tuple) => new SubstitutionRule(tuple.Item1, tuple.Item2, tuple.Item3, tuple.Item4);

```

```

45     public void OverrideMatchPatternOptions(RegexOptions options, TimeSpan matchTimeout) =>
46         ↳ MatchPattern = MatchPattern.OverrideOptions(options, matchTimeout);
47
48     public void OverridePathPatternOptions(RegexOptions options, TimeSpan matchTimeout) =>
49         ↳ PathPattern = PathPattern.OverrideOptions(options, matchTimeout);
50 }

```

./Platform.RegularExpressions.Transformer/TransformerCLI.cs

```

1  using System.Diagnostics;
2  using System.IO;
3  using System.Text;
4
5  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
6
7  namespace Platform.RegularExpressions.Transformer
8  {
9      public class TransformerCLI
10     {
11         private readonly ITransformer _transformer;
12
13         public TransformerCLI(ITransformer transformer) => _transformer = transformer;
14
15         public bool Run(string[] args, out string message)
16         {
17             message = "";
18             var sourcePath = GetArgOrDefault(args, 0);
19             if (!File.Exists(sourcePath))
20             {
21                 message = $"{sourcePath} file does not exist.";
22                 return false;
23             }
24             var targetPath = GetArgOrDefault(args, 1);
25             if (string.IsNullOrEmpty(targetPath))
26             {
27                 targetPath = Path.ChangeExtension(sourcePath, ".cpp");
28             }
29             else if ((Directory.Exists(targetPath) &&
30                 ↳ File.GetAttributes(targetPath).HasFlag(FileAttributes.Directory)) ||
31                 ↳ LooksLikeDirectoryPath(targetPath))
32             {
33                 targetPath = Path.Combine(targetPath,
34                     ↳ Path.ChangeExtension(Path.GetFileName(sourcePath), ".cpp"));
35             }
36             if (File.Exists(targetPath))
37             {
38                 var applicationPath = Process.GetCurrentProcess().MainModule.FileName;
39                 var targetFileLastUpdateDateTime = new FileInfo(targetPath).LastWriteTimeUtc;
40                 if (new FileInfo(sourcePath).LastWriteTimeUtc < targetFileLastUpdateDateTime &&
41                     ↳ new FileInfo(applicationPath).LastWriteTimeUtc <
42                     ↳ targetFileLastUpdateDateTime)
43                 {
44                     return true;
45                 }
46             }
47             File.WriteAllText(targetPath, _transformer.Transform(File.ReadAllText(sourcePath,
48                 ↳ Encoding.UTF8), new Context(sourcePath)), Encoding.UTF8);
49             message = $"{targetPath} file written.";
50             return true;
51         }
52
53         private static bool LooksLikeDirectoryPath(string targetPath) =>
54             ↳ targetPath.EndsWith(Path.DirectorySeparatorChar) ||
55             ↳ targetPath.EndsWith(Path.AltDirectorySeparatorChar);
56
57         private static string GetArgOrDefault(string[] args, int index) => args.Length > index ?
58             ↳ args[index] : null;
59     }
60 }

```

./Platform.RegularExpressions.Transformer/Transformer.cs

```

1  using System.Collections.Generic;
2
3  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
4
5  namespace Platform.RegularExpressions.Transformer
6  {

```

```

7     public class Transformer : ITransformer
8     {
9         private readonly IList<ISubstitutionRule> _substitutionRules;
10
11         public Transformer(IList<ISubstitutionRule> substitutionRules) => _substitutionRules =
            ↳ substitutionRules;
12
13         public string Transform(string source, IContext context)
14         {
15             var current = source;
16             for (var i = 0; i < _substitutionRules.Count; i++)
17             {
18                 var rule = _substitutionRules[i];
19                 var matchPattern = rule.MatchPattern;
20                 var substitutionPattern = rule.SubstitutionPattern;
21                 var pathPattern = rule.PathPattern;
22                 var maximumRepeatCount = rule.MaximumRepeatCount;
23                 if (pathPattern == null || pathPattern.IsMatch(context.Path))
24                 {
25                     var replaceCount = 0;
26                     do
27                     {
28                         current = matchPattern.Replace(current, substitutionPattern);
29                         if (++replaceCount > maximumRepeatCount)
30                         {
31                             break;
32                         }
33                     } while (matchPattern.IsMatch(current));
34                 }
35             }
36             return current;
37         }
38     }
39 }
40 }

```

./Platform.RegularExpressions.Transformer.Tests/obj/Release/netcoreapp3.0/Platform.RegularExpressions.Transformer.Tests

```

1  //-----
2  // <auto-generated>
3  //     Generated by the MSBuild WriteCodeFragment class.
4  // </auto-generated>
5  //-----
6
7  using System;
8  using System.Reflection;
9
10 [assembly:
    ↳ System.Reflection.AssemblyCompanyAttribute("Platform.RegularExpressions.Transformer.Tests")]
11 [assembly: System.Reflection.AssemblyConfigurationAttribute("Release")]
12 [assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]
13 [assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0")]
14 [assembly:
    ↳ System.Reflection.AssemblyProductAttribute("Platform.RegularExpressions.Transformer.Tests")]
15 [assembly:
    ↳ System.Reflection.AssemblyTitleAttribute("Platform.RegularExpressions.Transformer.Tests")]
16 [assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]

```

./Platform.RegularExpressions.Transformer.Tests/SubstitutionRuleTests.cs

```

1  using System.Text.RegularExpressions;
2  using Xunit;
3
4  namespace Platform.RegularExpressions.Transformer.Tests
5  {
6      public class SubstitutionRuleTests
7      {
8          [Fact]
9          public void OptionsOverrideTest()
10         {
11             SubstitutionRule rule = (new Regex(@"^s*?\#pragma[\\sa-zA-Z0-9\\/\r\n]+$"), "", null, 0);
12             Assert.Equal(RegexOptions.Compiled | RegexOptions.Multiline,
                ↳ rule.MatchPattern.Options);
13         }
14     }
15 }

```

Index

- ./Platform.RegularExpressions.Transformer.Tests/SubstitutionRuleTests.cs, 4
- ./Platform.RegularExpressions.Transformer.Tests/obj/Release/netcoreapp3.0/Platform.RegularExpressions.Transformer.Tests.AssemblyInfo.cs, 4
- ./Platform.RegularExpressions.Transformer/Context.cs, 1
- ./Platform.RegularExpressions.Transformer/IContext.cs, 1
- ./Platform.RegularExpressions.Transformer/ISubstitutionRule.cs, 1
- ./Platform.RegularExpressions.Transformer/ITransformer.cs, 1
- ./Platform.RegularExpressions.Transformer/RegexExtensions.cs, 1
- ./Platform.RegularExpressions.Transformer/SubstitutionRule.cs, 2
- ./Platform.RegularExpressions.Transformer/Transformer.cs, 3
- ./Platform.RegularExpressions.Transformer/TransformerCLI.cs, 3
- ./Platform.RegularExpressions.Transformer/obj/Release/netstandard2.1/Platform.RegularExpressions.Transformer.AssemblyInfo.cs, 1