

LinksPlatform's Platform.RegularExpressions.Transformer Class Library

1.1 ./csharp/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 }
```

1.2 ./csharp/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  }
```

1.3 ./csharp/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 }
```

1.4 ./csharp/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  }
```

1.5 ./csharp/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 }
```

1.6 ./csharp/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 =
12             ↳ RegexOptions.Compiled | RegexOptions.Multiline;
13         public static readonly RegexOptions DefaultPathPatternRegexOptions =
14             ↳ RegexOptions.Compiled | RegexOptions.Singleline;
15
16         public Regex MatchPattern { get; set; }
17
18         public string SubstitutionPattern { get; set; }
19
20         public Regex PathPattern { get; set; }
21
22         public int MaximumRepeatCount { get; set; }
23
24         public SubstitutionRule(Regex matchPattern, string substitutionPattern, Regex
25             ↳ pathPattern, int maximumRepeatCount, RegexOptions? matchPatternOptions,
26             ↳ RegexOptions? pathPatternOptions, TimeSpan? matchTimeout)
27         {
28             MatchPattern = matchPattern;
29             SubstitutionPattern = substitutionPattern;
30             PathPattern = pathPattern;
31             MaximumRepeatCount = maximumRepeatCount;
32             OverrideMatchPatternOptions(matchPatternOptions ?? matchPattern.Options,
33             ↳ matchTimeout ?? matchPattern.MatchTimeout);
34             OverridePathPatternOptions(pathPatternOptions ?? pathPattern.Options, matchTimeout
35             ↳ ?? pathPattern.MatchTimeout);
36         }
37
38         public SubstitutionRule(Regex matchPattern, string substitutionPattern, Regex
39             ↳ pathPattern, int maximumRepeatCount, bool useDefaultOptions) : this(matchPattern,
40             ↳ substitutionPattern, pathPattern, maximumRepeatCount, useDefaultOptions ?
41             ↳ DefaultMatchPatternRegexOptions : (RegexOptions?)null, useDefaultOptions ?
42             ↳ DefaultPathPatternRegexOptions : (RegexOptions?)null, useDefaultOptions ?
43             ↳ DefaultMatchTimeout : (TimeSpan?)null) { }
44
45         public SubstitutionRule(Regex matchPattern, string substitutionPattern, Regex
46             ↳ pathPattern, int maximumRepeatCount) : this(matchPattern, substitutionPattern,
47             ↳ pathPattern, maximumRepeatCount, true) { }
48
49         public SubstitutionRule(Regex matchPattern, string substitutionPattern, int
50             ↳ maximumRepeatCount) : this(matchPattern, substitutionPattern, null,
51             ↳ maximumRepeatCount) { }
52
53         public SubstitutionRule(Regex matchPattern, string substitutionPattern) :
54             ↳ this(matchPattern, substitutionPattern, null, 0) { }
55
56         public static implicit operator SubstitutionRule(ValueTuple<Regex, string> tuple) => new
57             ↳ SubstitutionRule(tuple.Item1, tuple.Item2);
58
59         public static implicit operator SubstitutionRule(ValueTuple<Regex, string, int> tuple)
60             ↳ => new SubstitutionRule(tuple.Item1, tuple.Item2, tuple.Item3);
61
62         public static implicit operator SubstitutionRule(ValueTuple<Regex, string, Regex, int>
63             ↳ tuple) => new SubstitutionRule(tuple.Item1, tuple.Item2, tuple.Item3, tuple.Item4);
64
65         public void OverrideMatchPatternOptions(RegexOptions options, TimeSpan matchTimeout) =>
66             ↳ MatchPattern = MatchPattern.OverrideOptions(options, matchTimeout);
67
68         public void OverridePathPatternOptions(RegexOptions options, TimeSpan matchTimeout) =>
69             ↳ PathPattern = PathPattern.OverrideOptions(options, matchTimeout);
70     }
71 }
```

1.7 ./csharp/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                     }
34                     while (matchPattern.IsMatch(current));
35                 }
36             }
37             return current;
38         }
39     }
40 }

```

1.8 ./csharp/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 = ChangeToTargetExtension(sourcePath);
28             }
29             else if (Directory.Exists(targetPath) &&
30                 ↳ File.GetAttributes(targetPath).HasFlag(FileAttributes.Directory))
31             {
32                 targetPath = Path.Combine(targetPath, GetTargetFileName(sourcePath));
33             }
34             else if (LooksLikeDirectoryPath(targetPath))
35             {
36                 Directory.CreateDirectory(targetPath);
37                 targetPath = Path.Combine(targetPath, GetTargetFileName(sourcePath));
38             }
39             if (File.Exists(targetPath))
40             {
41                 var applicationPath = Process.GetCurrentProcess().MainModule.FileName;
42                 var targetFileLastUpdateDateTime = new FileInfo(targetPath).LastWriteTimeUtc;

```

```

42         if (new FileInfo(sourcePath).LastWriteTimeUtc < targetFileLastUpdateDateTime &&
43             ↪ new FileInfo(applicationPath).LastWriteTimeUtc <
44             ↪ targetFileLastUpdateDateTime)
45         {
46             return true;
47         }
48     File.WriteAllText(targetPath, _transformer.Transform(File.ReadAllText(sourcePath,
49     ↪ Encoding.UTF8), new Context(sourcePath)), Encoding.UTF8);
50     message = $"{targetPath} file written.";
51     return true;
52 }
53
54 private static string GetTargetFileName(string sourcePath) =>
55     ↪ ChangeToTargetExtension(Path.GetFileName(sourcePath));
56
57 private static string ChangeToTargetExtension(string path) => Path.ChangeExtension(path,
58     ↪ ".cpp");
59
60 private static bool LooksLikeDirectoryPath(string targetPath) =>
61     ↪ targetPath.EndsWith(Path.DirectorySeparatorChar.ToString()) ||
62     ↪ targetPath.EndsWith(Path.AltDirectorySeparatorChar.ToString());
63
64 private static string GetArgOrDefault(string[] args, int index) => args.Length > index ?
65     ↪ args[index] : null;
66 }

```

1.9 ./csharp/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\/]+\$"), "", null, 0);
12             Assert.Equal(RegexOptions.Compiled | RegexOptions.Multiline,
13             ↪ rule.MatchPattern.Options);
14         }
15     }

```

Index

- ./csharp/Platform.RegularExpressions.Transformer.Tests/SubstitutionRuleTests.cs, 4
- ./csharp/Platform.RegularExpressions.Transformer/Context.cs, 1
- ./csharp/Platform.RegularExpressions.Transformer/IContext.cs, 1
- ./csharp/Platform.RegularExpressions.Transformer/ISubstitutionRule.cs, 1
- ./csharp/Platform.RegularExpressions.Transformer/ITransformer.cs, 1
- ./csharp/Platform.RegularExpressions.Transformer/RegexExtensions.cs, 1
- ./csharp/Platform.RegularExpressions.Transformer/SubstitutionRule.cs, 1
- ./csharp/Platform.RegularExpressions.Transformer/Transformer.cs, 2
- ./csharp/Platform.RegularExpressions.Transformer/TransformerCLI.cs, 3