

LinksPlatform's Platform.RegexTransformer Class Library

./Context.cs

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

./IContext.cs

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

./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.RegexTransformer
6  {
7      public interface ISubstitutionRule
8      {
9          Regex MatchPattern { get; }
10         string SubstitutionPattern { get; }
11         Regex PathPattern { get; }
12         int MaximumRepeatCount { get; }
13     }
14 }
```

./ITransformer.cs

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

./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.RegexTransformer
7  {
8      public static class RegexExtensions
9      {
10         public static Regex OverrideOptions(this Regex regex, RegexOptions options, TimeSpan
            ↳ matchTimeout) => new Regex(regex.ToString(), options, matchTimeout);
11     }
12 }
```

./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.RegexTransformer
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         OverrideMatchPatternOptions(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 }

```

./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.RegexTransformer
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 }

```

./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.RegexTransformer
6 {
7     public class Transformer : ITransformer
8     {
9         private readonly IList<ISubstitutionRule> _substitutionRules;
10
11         public Transformer(IList<ISubstitutionRule> substitutionRules) => _substitutionRules =
12             ↪ substitutionRules;
13
14         public string Transform(string source, IContext context)
15         {
16             var current = source;
17             for (var i = 0; i < _substitutionRules.Count; i++)
18             {
19                 var rule = _substitutionRules[i];
20                 var matchPattern = rule.MatchPattern;
21                 var substitutionPattern = rule.SubstitutionPattern;
22                 var pathPattern = rule.PathPattern;
23                 var maximumRepeatCount = rule.MaximumRepeatCount;
24                 if (pathPattern == null || pathPattern.IsMatch(context.Path))
25                 {
26                     var replaceCount = 0;
27                     do
28                     {
29                         current = matchPattern.Replace(current, substitutionPattern);
30                         if (++replaceCount > maximumRepeatCount)
31                         {
32                             break;
33                         }
34                     }
35                     while (true);
36                 }
37             }
38             return current;
39         }
40     }
41 }

```

```
33         }
34         while (matchPattern.IsMatch(current));
35     }
36 }
37 return current;
38 }
39 }
40 }
```

Index

./Context.cs, 1
./IContext.cs, 1
./ISubstitutionRule.cs, 1
./ITransformer.cs, 1
./RegexExtensions.cs, 1
./SubstitutionRule.cs, 1
./Transformer.cs, 3
./TransformerCLI.cs, 2