

LinksPlatform's Platform.Communication.Protocol.Lino Class Library

1.1 ./csharp/Platform.Communication.Protocol.Lino/ILinksGroupListExtensions.cs

```
1 using System.Collections.Generic;
2 using System.Runtime.CompilerServices;
3
4 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
5
6 namespace Platform.Communication.Protocol.Lino
7 {
8     public static class ILinksGroupListExtensions
9     {
10         [MethodImpl(MethodImplOptions.AggressiveInlining)]
11         public static List<Link> ToLinksList(this IList<LinksGroup> groups)
12         {
13             var list = new List<Link>();
14             for (var i = 0; i < groups.Count; i++)
15             {
16                 groups[i].AppendToLinksList(list);
17             }
18             return list;
19         }
20     }
21 }
```

1.2 ./csharp/Platform.Communication.Protocol.Lino/IListExtensions.cs

```
1 using Platform.Collections;
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Runtime.CompilerServices;
6
7 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
8
9 namespace Platform.Communication.Protocol.Lino
10 {
11     public static class IListExtensions
12     {
13         [MethodImpl(MethodImplOptions.AggressiveInlining)]
14         public static string Format(this IList<Link> links) => string.Join(Environment.NewLine,
15             ↪ links.Select(l => l.ToString()));
16
17         [MethodImpl(MethodImplOptions.AggressiveInlining)]
18         public static string Format(this IList<Link> links, bool lessParentheses)
19         {
20             if (lessParentheses == false)
21             {
22                 return links.Format();
23             }
24             else
25             {
26                 return string.Join(Environment.NewLine, links.Select(l =>
27                     ↪ l.ToString().TrimSingle('(').TrimSingle(')')));
28             }
29         }
30     }
31 }
```

1.3 ./csharp/Platform.Communication.Protocol.Lino/Link.cs

```
1 using System;
2 using System.Collections.Generic;
3 using System.Runtime.CompilerServices;
4 using System.Text;
5 using Platform.Collections;
6 using Platform.Collections.Lists;
7
8 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
9
10 namespace Platform.Communication.Protocol.Lino
11 {
12     public struct Link : IEquatable<Link>
13     {
14         public readonly string Id;
15
16         public readonly IList<Link> Values;
17
18         [MethodImpl(MethodImplOptions.AggressiveInlining)]
19         public Link(string id, IList<Link> values) => (Id, Values) = (id, values);
20
21         [MethodImpl(MethodImplOptions.AggressiveInlining)]
22         public Link(IList<Link> values) : this(null, values) { }
```

```

23 [MethodImpl(MethodImplOptions.AggressiveInlining)]
24 public Link(params Link[] values) : this(null, values) { }
25
26 [MethodImpl(MethodImplOptions.AggressiveInlining)]
27 public Link(string id) : this(id, null) { }
28
29 [MethodImpl(MethodImplOptions.AggressiveInlining)]
30 public override string ToString() => Values.IsNullOrEmpty() ? $"{Id}" :
31     ↪ GetLinkValuesString();
32
33 [MethodImpl(MethodImplOptions.AggressiveInlining)]
34 private string GetLinkValuesString() => Id == null ? $"{GetValuesString()}" :
35     ↪ $"{Id}: {GetValuesString()}";
36
37 [MethodImpl(MethodImplOptions.AggressiveInlining)]
38 public string GetValuesString()
39 {
40     var sb = new StringBuilder();
41     for (int i = 0; i < Values.Count; i++)
42     {
43         if (i > 0)
44         {
45             sb.Append(' ');
46         }
47         sb.Append(GetValueString(Values[i]));
48     }
49     return sb.ToString();
50 }
51
52 [MethodImpl(MethodImplOptions.AggressiveInlining)]
53 public Link Simplify()
54 {
55     if (Values.IsNullOrEmpty())
56     {
57         return this;
58     }
59     else if (Values.Count == 1)
60     {
61         return Values[0];
62     }
63     else
64     {
65         var newValues = new Link[Values.Count];
66         for (int i = 0; i < Values.Count; i++)
67         {
68             newValues[i] = Values[i].Simplify();
69         }
70         return new Link(Id, newValues);
71     }
72 }
73
74 [MethodImpl(MethodImplOptions.AggressiveInlining)]
75 public Link Combine(Link other) => new Link(this, other);
76
77 [MethodImpl(MethodImplOptions.AggressiveInlining)]
78 public static string GetValueString(Link value) => value.ToLinkOrIdString();
79
80 [MethodImpl(MethodImplOptions.AggressiveInlining)]
81 public string ToLinkOrIdString() => Values.IsNullOrEmpty() ? Id : ToString();
82
83 [MethodImpl(MethodImplOptions.AggressiveInlining)]
84 public static implicit operator Link(string value) => new Link(value);
85
86 [MethodImpl(MethodImplOptions.AggressiveInlining)]
87 public static implicit operator Link((string, IList<Link>) value) => new
88     ↪ Link(value.Item1, value.Item2);
89
90 [MethodImpl(MethodImplOptions.AggressiveInlining)]
91 public static implicit operator Link((Link, Link) value) => new Link(value.Item1,
92     ↪ value.Item2);
93
94 [MethodImpl(MethodImplOptions.AggressiveInlining)]
95 public static implicit operator Link((Link, Link, Link) value) => new Link(value.Item1,
96     ↪ value.Item2, value.Item3);
97
98 [MethodImpl(MethodImplOptions.AggressiveInlining)]
99 public override bool Equals(object obj) => obj is Link link ? Equals(link) : false;

```

```

96     [MethodImpl(MethodImplOptions.AggressiveInlining)]
97     public override int GetHashCode() => (Id, Values.GenerateHashCode()).GetHashCode();
98
99     [MethodImpl(MethodImplOptions.AggressiveInlining)]
100    public bool Equals(Link other) => Id == other.Id && Values.EqualTo(other.Values);
101
102    [MethodImpl(MethodImplOptions.AggressiveInlining)]
103    public static bool operator ==(Link left, Link right) => left.Equals(right);
104
105    [MethodImpl(MethodImplOptions.AggressiveInlining)]
106    public static bool operator !=(Link left, Link right) => !(left == right);
107
108    }
109 }

```

1.4 ./csharp/Platform.Communication.Protocol.Lino/LinksGroup.cs

```

1  using System;
2  using System.Collections.Generic;
3  using System.Runtime.CompilerServices;
4  using Platform.Collections;
5  using Platform.Collections.Lists;
6
7  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
8
9  namespace Platform.Communication.Protocol.Lino
10 {
11     public struct LinksGroup : IEquatable<LinksGroup>
12     {
13         public Link Link
14         {
15             [MethodImpl(MethodImplOptions.AggressiveInlining)]
16             get;
17             [MethodImpl(MethodImplOptions.AggressiveInlining)]
18             set;
19         }
20
21         public IList<LinksGroup> Groups
22         {
23             [MethodImpl(MethodImplOptions.AggressiveInlining)]
24             get;
25             [MethodImpl(MethodImplOptions.AggressiveInlining)]
26             set;
27         }
28
29         [MethodImpl(MethodImplOptions.AggressiveInlining)]
30         public LinksGroup(Link link, IList<LinksGroup> groups)
31         {
32             Link = link;
33             Groups = groups;
34         }
35
36         [MethodImpl(MethodImplOptions.AggressiveInlining)]
37         public LinksGroup(Link link) : this(link, null) { }
38
39         [MethodImpl(MethodImplOptions.AggressiveInlining)]
40         public static implicit operator List<Link>(LinksGroup value) => value.ToLinksList();
41
42         [MethodImpl(MethodImplOptions.AggressiveInlining)]
43         public List<Link> ToLinksList()
44         {
45             var list = new List<Link>();
46             AppendToLinksList(list);
47             return list;
48         }
49
50         [MethodImpl(MethodImplOptions.AggressiveInlining)]
51         public void AppendToLinksList(List<Link> list) => AppendToLinksList(list, Link, this);
52
53         [MethodImpl(MethodImplOptions.AggressiveInlining)]
54         public static void AppendToLinksList(List<Link> list, Link dependency, LinksGroup group)
55         {
56             list.Add(dependency);
57             var groups = group.Groups;
58             if (!groups.IsNullOrEmpty())
59             {
60                 for (int i = 0; i < groups.Count; i++)
61                 {
62                     var innerGroup = groups[i];
63                     AppendToLinksList(list, dependency.Combine(innerGroup.Link), innerGroup);
64                 }
65             }
66         }
67     }
68 }

```

```

65     }
66 }
67
68 [MethodImpl(MethodImplOptions.AggressiveInlining)]
69 public override bool Equals(object obj) => obj is LinksGroup linksGroup ?
    ↳ Equals(linksGroup) : false;
70
71 [MethodImpl(MethodImplOptions.AggressiveInlining)]
72 public override int GetHashCode() => (Link, Groups.GenerateHashCode()).GetHashCode();
73
74 [MethodImpl(MethodImplOptions.AggressiveInlining)]
75 public bool Equals(LinksGroup other) => Link == other.Link &&
    ↳ Groups.EqualTo(other.Groups);
76
77 [MethodImpl(MethodImplOptions.AggressiveInlining)]
78 public static bool operator ==(LinksGroup left, LinksGroup right) => left.Equals(right);
79
80 [MethodImpl(MethodImplOptions.AggressiveInlining)]
81 public static bool operator !=(LinksGroup left, LinksGroup right) => !(left == right);
82 }
83 }

```

1.5 ./csharp/Platform.Communication.Protocol.Lino/__.cs

```

1  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
2
3  using System.Runtime.CompilerServices;
4
5  namespace Platform.Communication.Protocol.Lino
6  {
7      public struct _
8      {
9          public readonly Link Link;
10
11          [MethodImpl(MethodImplOptions.AggressiveInlining)]
12          public _(Link id) => Link = id;
13
14          [MethodImpl(MethodImplOptions.AggressiveInlining)]
15          public static implicit operator _(Link value) => new _(value);
16
17          [MethodImpl(MethodImplOptions.AggressiveInlining)]
18          public static implicit operator _((string id) => new _(id);
19
20          [MethodImpl(MethodImplOptions.AggressiveInlining)]
21          public static implicit operator _((string, Link) value) => new Link(value.Item1, new
    ↳ Link[] { value.Item2 });
22
23          [MethodImpl(MethodImplOptions.AggressiveInlining)]
24          public static implicit operator _((string, Link, Link) value) => new Link(value.Item1,
    ↳ new Link[] { value.Item2, value.Item3 });
25
26          [MethodImpl(MethodImplOptions.AggressiveInlining)]
27          public static implicit operator _((string, Link, Link, Link) value) => new
    ↳ Link(value.Item1, new Link[] { value.Item2, value.Item3, value.Item4 });
28
29          [MethodImpl(MethodImplOptions.AggressiveInlining)]
30          public static implicit operator Link(_ value) => value.Link;
31      }
32 }

```

1.6 ./csharp/Platform.Communication.Protocol.Lino.Tests/ParserTests.cs

```

1  using Xunit;
2
3  namespace Platform.Communication.Protocol.Lino.Tests
4  {
5      public static class ParserTests
6      {
7          [Fact]
8          public static void SingleLinkTest()
9          {
10              var source = @"(address: source target)";
11              var parser = new Parser();
12              var links = parser.Parse(source);
13              var target = links.Format();
14              Assert.Equal(source, target);
15          }
16
17          [Fact]
18          public static void TwoLinksTest()
19          {

```

```

20         var source = @"(first: x y)
21 (second: a b)";
22         var parser = new Parser();
23         var links = parser.Parse(source);
24         var target = links.Format();
25         Assert.Equal(source, target);
26     }
27
28     [Fact]
29     public static void ParseAndStringifyTest()
30     {
31         var source = @"(papa (lovesMama: loves mama))
32 (son lovesMama)
33 (daughter lovesMama)
34 (all (love mama))";
35         var parser = new Parser();
36         var links = parser.Parse(source);
37         var target = links.Format();
38         Assert.Equal(source, target);
39     }
40
41     [Fact]
42     public static void BugTest()
43     {
44         var source = @"(ignore conan-center-index repository)";
45         var links = (new Platform.Communication.Protocol.Lino.Parser()).Parse(source);
46         var target = links.Format();
47         Assert.Equal(source, target);
48     }
49
50     [Fact]
51     public static void ParseAndStringifyWithLessParenthesesTest()
52     {
53         var source = @"lovesMama: loves mama
54 papa lovesMama
55 son lovesMama
56 daughter lovesMama
57 all (love mama)";
58         var parser = new Parser();
59         var links = parser.Parse(source);
60         var target = links.Format(lessParentheses: true);
61         Assert.Equal(source, target);
62     }
63
64     [Fact]
65     public static void SignificantWhitespaceTest()
66     {
67         var source = @"
68 users
69     user1
70         id
71             43
72         name
73             first
74                 John
75             last
76                 Williams
77         location
78             New York
79         age
80             23
81     user2
82         id
83             56
84         name
85             first
86                 Igor
87             middle
88                 Petrovich
89             last
90                 Ivanov
91         location
92             Moscow
93         age
94             20";
95         var target = @"(users)
96 (users user1)
97 ((users user1) id)
98 (((users user1) id) 43)
99 ((users user1) name)
100 (((users user1) name) first)

```

```

101 (((users user1) name) first) John)
102 (((users user1) name) last)
103 (((users user1) name) last) Williams)
104 ((users user1) location)
105 (((users user1) location) (New York))
106 ((users user1) age)
107 (((users user1) age) 23)
108 (users user2)
109 ((users user2) id)
110 (((users user2) id) 56)
111 ((users user2) name)
112 (((users user2) name) first)
113 (((users user2) name) first) Igor)
114 (((users user2) name) middle)
115 (((users user2) name) middle) Petrovich)
116 ((users user2) name) last)
117 (((users user2) name) last) Ivanov)
118 ((users user2) location)
119 (((users user2) location) Moscow)
120 ((users user2) age)
121 (((users user2) age) 20)";
122         var parser = new Parser();
123         var links = parser.Parse(source);
124         var formattedLinks = links.Format();
125         Assert.Equal(target, formattedLinks);
126     }
127 }
128 }

```

1.7 ./csharp/Platform.Communication.Protocol.Lino.Tests/TupleTests.cs

```

1 using System.Collections.Generic;
2 using Xunit;
3
4 namespace Platform.Communication.Protocol.Lino.Tests
5 {
6     public class TupleTests
7     {
8         [Fact]
9         public void TupleToLinkTest()
10        {
11            var source = @"(papa (lovesMama: loves mama))
12(son lovesMama)
13(daughter lovesMama)
14(all (love mama))";
15            var parser = new Parser();
16            var links = parser.Parse(source);
17            var targetFromString = links.Format();
18
19            IList<Link> constructedLinks = new List<Link>()
20            {
21                ("papa", (_)("lovesMama", "loves", "mama")),
22                ("son", "lovesMama"),
23                ("daughter", "lovesMama"),
24                ("all", ("love", "mama")),
25            };
26            var targetFromTuples = constructedLinks.Format();
27            Assert.Equal(targetFromString, targetFromTuples);
28        }
29    }
30 }

```

Index

./csharp/Platform.Communication.Protocol.Lino.Tests/ParserTests.cs, 4
./csharp/Platform.Communication.Protocol.Lino.Tests/TupleTests.cs, 6
./csharp/Platform.Communication.Protocol.Lino/ILinksGroupListExtensions.cs, 1
./csharp/Platform.Communication.Protocol.Lino/IListExtensions.cs, 1
./csharp/Platform.Communication.Protocol.Lino/Link.cs, 1
./csharp/Platform.Communication.Protocol.Lino/LinksGroup.cs, 3
./csharp/Platform.Communication.Protocol.Lino/_.cs, 4