

LinksPlatform's Platform.Data.Doublets.Xml Class Library

1.1 ./csharp/Platform.Data.Doublets.Xml/DefaultXmlStorage.cs

```
1  using System.Collections.Generic;
2  using Platform.Numbers;
3  using Platform.Data.Numbers.Raw;
4  using Platform.Data.Doublets;
5  using Platform.Data.Doublets.Sequences.Converters;
6  using Platform.Data.Doublets.Sequences.Frequencies.Cache;
7  using Platform.Data.Doublets.Sequences.Frequencies.Counters;
8  using Platform.Data.Doublets.Sequences.Indexes;
9  using Platform.Data.Doublets.Unicode;
10
11 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
12
13 namespace Platform.Data.Doublets.Xml
14 {
15     /// <summary>
16     /// <para>
17     /// Represents the default xml storage.
18     /// </para>
19     /// <para></para>
20     /// </summary>
21     /// <seealso cref="IXmlStorage{TLink}" />
22     public class DefaultXmlStorage<TLink> : IXmlStorage<TLink>
23     {
24         private static readonly TLink _zero = default;
25         private static readonly TLink _one = Arithmetic.Increment(_zero);
26         private readonly StringToUnicodeSequenceConverter<TLink>
27             ↪ _stringToUnicodeSequenceConverter;
28         private readonly ILinks<TLink> _links;
29         private TLink _unicodeSymbolMarker;
30         private TLink _unicodeSequenceMarker;
31         private TLink _elementMarker;
32         private TLink _textElementMarker;
33         private TLink _documentMarker;
34
35         private class Unindex : ISequenceIndex<TLink>
36         {
37             public bool Add(IList<TLink> sequence) => true;
38
39             public bool MightContain(IList<TLink> sequence) => true;
40         }
41
42         /// <summary>
43         /// <para>
44         /// Initializes a new <see cref="DefaultXmlStorage" /> instance.
45         /// </para>
46         /// <para></para>
47         /// </summary>
48         /// <param name="links">
49         /// <para>A links.</para>
50         /// <para></para>
51         /// </param>
52         /// <param name="indexSequenceBeforeCreation">
53         /// <para>A index sequence before creation.</para>
54         /// <para></para>
55         /// </param>
56         /// <param name="frequenciesCache">
57         /// <para>A frequencies cache.</para>
58         /// <para></para>
59         /// </param>
60         public DefaultXmlStorage(ILinks<TLink> links, bool indexSequenceBeforeCreation,
61             ↪ LinkFrequenciesCache<TLink> frequenciesCache)
62         {
63             var linkToItsFrequencyNumberConverter = new
64                 ↪ FrequenciesCacheBasedLinkToItsFrequencyNumberConverter<TLink>(frequenciesCache);
65             var sequenceToItsLocalElementLevelsConverter = new
66                 ↪ SequenceToItsLocalElementLevelsConverter<TLink>(links,
67                 ↪ linkToItsFrequencyNumberConverter);
68             var optimalVariantConverter = new OptimalVariantConverter<TLink>(links,
69                 ↪ sequenceToItsLocalElementLevelsConverter);
70             InitConstants(links);
71             var charToUnicodeSymbolConverter = new CharToUnicodeSymbolConverter<TLink>(links,
72                 ↪ new AddressToRawNumberConverter<TLink>(), _unicodeSymbolMarker);
73             var index = indexSequenceBeforeCreation ? new
74                 ↪ CachedFrequencyIncrementingSequenceIndex<TLink>(frequenciesCache) :
75                 ↪ (ISequenceIndex<TLink>)new Unindex();
76         }
77     }
78 }
```

```

67     _stringToUnicodeSequenceConverter = new
        ↳ StringToUnicodeSequenceConverter<TLink>(links, charToUnicodeSymbolConverter,
68         ↳ index, optimalVariantConverter, _unicodeSequenceMarker);
69     _links = links;
70 }
71 public DefaultXmlStorage(ILinks<TLink> links, bool indexSequenceBeforeCreation = false) :
72     this(links, indexSequenceBeforeCreation,
73         new LinkFrequenciesCache<TLink>(links,
74             new TotalSequenceSymbolFrequencyCounter<TLink>(links))) { }
75
76 private void InitConstants(ILinks<TLink> links)
77 {
78     var markerIndex = _one;
79     var meaningRoot = links.GetOrCreate(markerIndex, markerIndex);
80     _unicodeSymbolMarker = links.GetOrCreate(meaningRoot, Arithmetic.Increment(ref
        ↳ markerIndex));
81     _unicodeSequenceMarker = links.GetOrCreate(meaningRoot, Arithmetic.Increment(ref
        ↳ markerIndex));
82     _elementMarker = links.GetOrCreate(meaningRoot, Arithmetic.Increment(ref
        ↳ markerIndex));
83     _textElementMarker = links.GetOrCreate(meaningRoot, Arithmetic.Increment(ref
        ↳ markerIndex));
84     _documentMarker = links.GetOrCreate(meaningRoot, Arithmetic.Increment(ref
        ↳ markerIndex));
85 }
86
87 /// <summary>
88 /// <para>
89 /// Creates the document using the specified name.
90 /// </para>
91 /// <para></para>
92 /// </summary>
93 /// <param name="name">
94 /// <para>The name.</para>
95 /// <para></para>
96 /// </param>
97 /// <returns>
98 /// <para>The link</para>
99 /// <para></para>
100 /// </returns>
101 public TLink CreateDocument(string name) => Create(_documentMarker, name);
102
103 /// <summary>
104 /// <para>
105 /// Creates the element using the specified name.
106 /// </para>
107 /// <para></para>
108 /// </summary>
109 /// <param name="name">
110 /// <para>The name.</para>
111 /// <para></para>
112 /// </param>
113 /// <returns>
114 /// <para>The link</para>
115 /// <para></para>
116 /// </returns>
117 public TLink CreateElement(string name) => Create(_elementMarker, name);
118
119 /// <summary>
120 /// <para>
121 /// Creates the text element using the specified content.
122 /// </para>
123 /// <para></para>
124 /// </summary>
125 /// <param name="content">
126 /// <para>The content.</para>
127 /// <para></para>
128 /// </param>
129 /// <returns>
130 /// <para>The link</para>
131 /// <para></para>
132 /// </returns>
133 public TLink CreateTextElement(string content) => Create(_textElementMarker, content);
134
135 private TLink Create(TLink marker, string content) => _links.GetOrCreate(marker,
    ↳ _stringToUnicodeSequenceConverter.Convert(content));
136

```

```

137     /// <summary>
138     /// <para>
139     /// Attaches the element to parent using the specified element to attach.
140     /// </para>
141     /// <para></para>
142     /// </summary>
143     /// <param name="elementToAttach">
144     /// <para>The element to attach.</para>
145     /// <para></para>
146     /// </param>
147     /// <param name="parent">
148     /// <para>The parent.</para>
149     /// <para></para>
150     /// </param>
151     public void AttachElementToParent(TLink elementToAttach, TLink parent) =>
152         ↪ _links.GetOrCreate(parent, elementToAttach);
153
154     /// <summary>
155     /// <para>
156     /// Gets the document using the specified name.
157     /// </para>
158     /// <para></para>
159     /// </summary>
160     /// <param name="name">
161     /// <para>The name.</para>
162     /// <para></para>
163     /// </param>
164     /// <returns>
165     /// <para>The link</para>
166     /// <para></para>
167     /// </returns>
168     public TLink GetDocument(string name) => Get(_documentMarker, name);
169
170     /// <summary>
171     /// <para>
172     /// Gets the text element using the specified content.
173     /// </para>
174     /// <para></para>
175     /// </summary>
176     /// <param name="content">
177     /// <para>The content.</para>
178     /// <para></para>
179     /// </param>
180     /// <returns>
181     /// <para>The link</para>
182     /// <para></para>
183     /// </returns>
184     public TLink GetTextElement(string content) => Get(_textElementMarker, content);
185
186     /// <summary>
187     /// <para>
188     /// Gets the element using the specified name.
189     /// </para>
190     /// <para></para>
191     /// </summary>
192     /// <param name="name">
193     /// <para>The name.</para>
194     /// <para></para>
195     /// </param>
196     /// <returns>
197     /// <para>The link</para>
198     /// <para></para>
199     /// </returns>
200     public TLink GetElement(string name) => Get(_elementMarker, name);
201
202     private TLink Get(TLink marker, string content) => _links.SearchOrDefault(marker,
203         ↪ _stringToUnicodeSequenceConverter.Convert(content));
204
205     /// <summary>
206     /// <para>
207     /// Gets the children using the specified parent.
208     /// </para>
209     /// <para></para>
210     /// </summary>
211     /// <param name="parent">
212     /// <para>The parent.</para>
213     /// <para></para>
214     /// </param>

```

```

213     /// <returns>
214     /// <para>The childrens.</para>
215     /// <para></para>
216     /// </returns>
217     public IList<TLink> GetChildren(TLink parent)
218     {
219         List<TLink> childrens = new List<TLink>();
220         _links.Each((link) => {
221             childrens.Add(_links.GetTarget(link));
222             return this._links.Constants.Continue;
223         }, new Link<TLink>(_links.Constants.Any, parent, _links.Constants.Any));
224         return childrens;
225     }
226 }
227 }

```

1.2 ./csharp/Platform.Data.Doublets.Xml/ICommandLineInterface.cs

```

1  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
2
3  namespace Platform.Data.Doublets.Xml
4  {
5      /// <summary>
6      /// <para>
7      /// Defines the command line interface.
8      /// </para>
9      /// <para></para>
10     /// </summary>
11     public interface ICommandLineInterface
12     {
13         /// <summary>
14         /// <para>
15         /// Runs the args.
16         /// </para>
17         /// <para></para>
18         /// </summary>
19         /// <param name="args">
20         /// <para>The args.</para>
21         /// <para></para>
22         /// </param>
23         void Run(params string[] args);
24     }
25 }

```

1.3 ./csharp/Platform.Data.Doublets.Xml/IXmlStorage.cs

```

1  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
2
3  using System.Collections.Generic;
4
5  namespace Platform.Data.Doublets.Xml
6  {
7      /// <summary>
8      /// <para>
9      /// Defines the xml storage.
10     /// </para>
11     /// <para></para>
12     /// </summary>
13     public interface IXmlStorage<TLink>
14     {
15         /// <summary>
16         /// <para>
17         /// Creates the document using the specified name.
18         /// </para>
19         /// <para></para>
20         /// </summary>
21         /// <param name="name">
22         /// <para>The name.</para>
23         /// <para></para>
24         /// </param>
25         /// <returns>
26         /// <para>The link</para>
27         /// <para></para>
28         /// </returns>
29         TLink CreateDocument(string name);
30
31         /// <summary>
32         /// <para>
33         /// Creates the element using the specified name.
34         /// </para>

```

```

35     /// <para></para>
36     /// </summary>
37     /// <param name="name">
38     /// <para>The name.</para>
39     /// <para></para>
40     /// </param>
41     /// <returns>
42     /// <para>The link</para>
43     /// <para></para>
44     /// </returns>
45     TLink CreateElement(string name);
46
47     /// <summary>
48     /// <para>
49     /// Creates the text element using the specified content.
50     /// </para>
51     /// <para></para>
52     /// </summary>
53     /// <param name="content">
54     /// <para>The content.</para>
55     /// <para></para>
56     /// </param>
57     /// <returns>
58     /// <para>The link</para>
59     /// <para></para>
60     /// </returns>
61     TLink CreateTextElement(string content);
62
63     /// <summary>
64     /// <para>
65     /// Gets the document using the specified name.
66     /// </para>
67     /// <para></para>
68     /// </summary>
69     /// <param name="name">
70     /// <para>The name.</para>
71     /// <para></para>
72     /// </param>
73     /// <returns>
74     /// <para>The link</para>
75     /// <para></para>
76     /// </returns>
77     TLink GetDocument(string name);
78
79     /// <summary>
80     /// <para>
81     /// Gets the element using the specified name.
82     /// </para>
83     /// <para></para>
84     /// </summary>
85     /// <param name="name">
86     /// <para>The name.</para>
87     /// <para></para>
88     /// </param>
89     /// <returns>
90     /// <para>The link</para>
91     /// <para></para>
92     /// </returns>
93     TLink GetElement(string name);
94
95     /// <summary>
96     /// <para>
97     /// Gets the text element using the specified content.
98     /// </para>
99     /// <para></para>
100    /// </summary>
101    /// <param name="content">
102    /// <para>The content.</para>
103    /// <para></para>
104    /// </param>
105    /// <returns>
106    /// <para>The link</para>
107    /// <para></para>
108    /// </returns>
109    TLink GetTextElement(string content);
110
111    /// <summary>
112    /// <para>

```

```

113     /// Gets the children using the specified parent.
114     /// </para>
115     /// <para></para>
116     /// </summary>
117     /// <param name="parent">
118     /// <para>The parent.</para>
119     /// <para></para>
120     /// </param>
121     /// <returns>
122     /// <para>A list of i list t link</para>
123     /// <para></para>
124     /// </returns>
125     IList<TLink> GetChildren(TLink parent);
126
127     /// <summary>
128     /// <para>
129     /// Attaches the element to parent using the specified element to attach.
130     /// </para>
131     /// <para></para>
132     /// </summary>
133     /// <param name="elementToAttach">
134     /// <para>The element to attach.</para>
135     /// <para></para>
136     /// </param>
137     /// <param name="parent">
138     /// <para>The parent.</para>
139     /// <para></para>
140     /// </param>
141     void AttachElementToParent(TLink elementToAttach, TLink parent);
142 }
143 }

```

1.4 ./csharp/Platform.Data.Doublets.Xml/XmlElementContext.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.Data.Doublets.Xml
6 {
7     /// <summary>
8     /// <para>
9     /// Represents the xml element context.
10    /// </para>
11    /// <para></para>
12    /// </summary>
13    internal class XmlElementContext
14    {
15        public readonly Dictionary<string, int> ChildrenNamesCounts;
16        public int TotalChildren;
17
18        public XmlElementContext() => ChildrenNamesCounts = new Dictionary<string, int>();
19
20        public void IncrementChildNameCount(string name)
21        {
22            if (ChildrenNamesCounts.TryGetValue(name, out int count))
23            {
24                ChildrenNamesCounts[name] = count + 1;
25            }
26            else
27            {
28                ChildrenNamesCounts[name] = 0;
29            }
30            TotalChildren++;
31        }
32    }
33 }

```

1.5 ./csharp/Platform.Data.Doublets.Xml/XmlElementCounter.cs

```

1 using System;
2 using System.Collections.Generic;
3 using System.Threading;
4 using System.Threading.Tasks;
5 using System.Xml;
6 using System.Linq;
7 using Platform.Exceptions;
8 using Platform.IO;
9
10 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
11

```

```

12 namespace Platform.Data.Doublets.Xml
13 {
14     /// <summary>
15     /// <para>
16     /// Represents the xml element counter.
17     /// </para>
18     /// <para></para>
19     /// </summary>
20     public class XmlElementCounter
21     {
22         /// <summary>
23         /// <para>
24         /// Initializes a new <see cref="XmlElementCounter"/> instance.
25         /// </para>
26         /// <para></para>
27         /// </summary>
28         public XmlElementCounter() { }
29
30         /// <summary>
31         /// <para>
32         /// Counts the file.
33         /// </para>
34         /// <para></para>
35         /// </summary>
36         /// <param name="file">
37         /// <para>The file.</para>
38         /// <para></para>
39         /// </param>
40         /// <param name="elementName">
41         /// <para>The element name.</para>
42         /// <para></para>
43         /// </param>
44         /// <param name="token">
45         /// <para>The token.</para>
46         /// <para></para>
47         /// </param>
48         public Task Count(string file, string elementName, CancellationToken token)
49         {
50             return Task.Factory.StartNew(() =>
51             {
52                 try
53                 {
54                     var context = new RootElementContext();
55                     using (var reader = XmlReader.Create(file))
56                     {
57                         Count(reader, elementName, token, context);
58                     }
59                     Console.WriteLine($"Total elements with specified name:
60                                     ↳ {context.TotalElements}, total content length:
61                                     ↳ {context.TotalContentsLength}.");
62                 }
63                 catch (Exception ex)
64                 {
65                     Console.WriteLine(ex.ToStringWithAllInnerExceptions());
66                 }
67             }, token);
68
69             private void Count(XmlReader reader, string elementNameToCount, CancellationToken token,
70                               XmlElementContext context)
71             {
72                 var rootContext = (RootElementContext)context;
73                 var parentContexts = new Stack<XmlElementContext>();
74                 var elements = new Stack<string>(); // Path
75                 // TODO: If path was loaded previously, skip it.
76                 while (reader.Read())
77                 {
78                     if (token.IsCancellationRequested)
79                     {
80                         return;
81                     }
82                     switch (reader.NodeType)
83                     {
84                         case XmlNodeType.Element:
85                             var elementName = reader.Name;
86                             context.IncrementChildNameCount(elementName);
87                             elementName =
88                                 $"{elementName}[{context.ChildrenNamesCounts[elementName]}]";

```

```

86         if (!reader.IsEmptyElement)
87         {
88             elements.Push(elementName);
89             ConsoleHelpers.Debug("{0} starting...", elements.Count <= 20 ?
            ↪ ToXPath(elements) : elementName); // XPath
90             parentContexts.Push(context);
91             context = new XmlElementContext();
92         }
93         else
94         {
95             ConsoleHelpers.Debug("{0} finished.", elementName);
96         }
97         break;
98
99     case XmlNodeType.EndElement:
100         ConsoleHelpers.Debug("{0} finished.", elements.Count <= 20 ?
            ↪ ToXPath(elements) : elements.Peek()); // XPath
101         var topElement = elements.Pop();
102         // Restoring scope
103         context = parentContexts.Pop();
104         if (topElement.StartsWith(elementNameToCount))
105         {
106             rootContext.TotalElements++;
107             // TODO: Check for 0x00 part/symbol at 198102797 line and 13
            ↪ position.
108             //if (rootContext.TotalPages > 3490000)
109             //    selfCancel = true;
110             if (context.ChildrenNamesCounts[elementNameToCount] % 10000 == 0)
111             {
112                 Console.WriteLine(topElement);
113             }
114         }
115         break;
116
117     case XmlNodeType.Text:
118         ConsoleHelpers.Debug("Starting text element...");
119         var content = reader.Value;
120         rootContext.TotalContentsLength += (ulong)content.Length;
121         ConsoleHelpers.Debug($"Content length is: {content.Length}");
122         ConsoleHelpers.Debug("Text element finished.");
123         break;
124     }
125 }
126 }
127
128 private string ToXPath(Stack<string> path) => string.Join("/", path.Reverse());
129
130 private class RootElementContext : XmlElementContext
131 {
132     public ulong TotalElements;
133     public ulong TotalContentsLength;
134 }
135 }
136 }

```

1.6 ./csharp/Platform.Data.Doublets.Xml/XmlElementCounterCLI.cs

```

1  using System;
2  using System.IO;
3  using Platform.IO;
4
5  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
6
7  namespace Platform.Data.Doublets.Xml
8  {
9      /// <summary>
10     /// <para>
11     /// Represents the xml element counter cli.
12     /// </para>
13     /// <para></para>
14     /// </summary>
15     /// <seealso cref="ICommandLineInterface"/>
16     public class XmlElementCounterCLI : ICommandLineInterface
17     {
18         /// <summary>
19         /// <para>
20         /// Runs the args.
21         /// </para>
22         /// <para></para>
23         /// </summary>

```



```

24     /// <param name="args">
25     /// <para>The args.</para>
26     /// <para></para>
27     /// </param>
28     public void Run(params string[] args)
29     {
30         var file = ConsoleHelpers.GetOrReadArgument(0, "Xml file", args);
31         var elementName = ConsoleHelpers.GetOrReadArgument(1, "Element name to count", args);
32         if (!File.Exists(file))
33         {
34             Console.WriteLine("Entered xml file does not exists.");
35         }
36         else if (string.IsNullOrEmpty(elementName))
37         {
38             Console.WriteLine("Entered element name is empty.");
39         }
40         else
41         {
42             using (var cancellation = new ConsoleCancellation())
43             {
44                 Console.WriteLine("Press CTRL+C to stop.");
45                 new XmlElementCounter().Count(file, elementName, cancellation.Token).Wait();
46             }
47         }
48     }
49 }
50 }

```

1.7 ./csharp/Platform.Data.Doublets.Xml/XmlExporter.cs

```

1  using System;
2  using System.Linq;
3  using System.Collections.Generic;
4  using System.Threading;
5  using System.Threading.Tasks;
6  using System.Xml;
7  using Platform.Exceptions;
8
9  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
10
11 namespace Platform.Data.Doublets.Xml
12 {
13     /// <summary>
14     /// <para>
15     /// Represents the xml exporter.
16     /// </para>
17     /// <para></para>
18     /// </summary>
19     public class XmlExporter<TLink>
20     {
21         private readonly IXmlStorage<TLink> _storage;
22
23         /// <summary>
24         /// <para>
25         /// Initializes a new <see cref="XmlExporter"/> instance.
26         /// </para>
27         /// <para></para>
28         /// </summary>
29         /// <param name="storage">
30         /// <para>A storage.</para>
31         /// <para></para>
32         /// </param>
33         public XmlExporter(IXmlStorage<TLink> storage) => _storage = storage;
34
35         /// <summary>
36         /// <para>
37         /// Exports the document name.
38         /// </para>
39         /// <para></para>
40         /// </summary>
41         /// <param name="documentName">
42         /// <para>The document name.</para>
43         /// <para></para>
44         /// </param>
45         /// <param name="fileName">
46         /// <para>The file name.</para>
47         /// <para></para>
48         /// </param>
49         /// <param name="token">
50         /// <para>The token.</para>

```

```

51     /// <para></para>
52     /// </param>
53     public Task Export(string documentName, string fileName, CancellationToken token)
54     {
55         return Task.Factory.StartNew(() =>
56         {
57             try
58             {
59                 var document = _storage.GetDocument(documentName);
60                 using (var writer = XmlWriter.Create(fileName))
61                 {
62                     Write(writer, token, new ElementContext(document));
63                 }
64             }
65             catch (Exception ex)
66             {
67                 Console.WriteLine(ex.ToStringWithAllInnerExceptions());
68             }
69         }, token);
70     }
71
72     private void Write(XmlWriter writer, CancellationToken token, ElementContext context)
73     {
74         var parentContexts = new Stack<ElementContext>();
75         var elements = new Stack<string>(); // Path
76                                     // TODO: If path was loaded previously, skip it.
77         foreach(TLink lvl in _storage.GetChildren(parent: context.Parent))
78         {
79             Write(writer: writer, token: token, context: new ElementContext(lvl));
80         }
81     }
82
83     private class ElementContext : XmlElementContext
84     {
85         public readonly TLink Parent;
86         public ElementContext(TLink parent) => Parent = parent;
87     }
88
89 }
90 }

```

1.8 ./csharp/Platform.Data.Doublets.Xml/XmlExporterCLI.cs

```

1  using System;
2  using System.IO;
3  using Platform.IO;
4  using Platform.Data.Doublets.Memory.United.Generic;
5
6  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
7
8  namespace Platform.Data.Doublets.Xml
9  {
10     public class XmlExporterCLI : ICommandLineInterface
11     {
12         public void Run(params string[] args)
13         {
14             var linksFile = ConsoleHelpers.GetOrReadArgument(0, "Links file", args);
15             var exportFile = ConsoleHelpers.GetOrReadArgument(1, "Xml file", args);
16
17             if (File.Exists(exportFile))
18                 Console.WriteLine("Entered xml file does already exists.");
19             else
20             {
21                 using var cancellation = new ConsoleCancellation();
22                 var linksConstants = new LinksConstants<ulong>(enableExternalReferencesSupport:
23                     ↪ true);
24                 using var memoryAdapter = new UnitedMemoryLinks<uint>(linksFile);
25                 Console.WriteLine("Press CTRL+C to stop.");
26                 var links = memoryAdapter.DecorateWithAutomaticUniquenessAndUsagesResolution();
27                 if (!cancellation.NotRequested) return;
28                 var storage = new DefaultXmlStorage<uint>(links);
29                 var exporter = new XmlExporter<uint>(storage);
30                 exporter.Export(linksFile, exportFile, cancellation.Token).Wait();
31             }
32         }
33     }
34 }

```

1.9 ./csharp/Platform.Data.Doublets.Xml/XmlImporter.cs

```
1 using System;
2 using System.Linq;
3 using System.Collections.Generic;
4 using System.Threading;
5 using System.Threading.Tasks;
6 using System.Xml;
7 using Platform.Exceptions;
8 using Platform.Collections;
9 using Platform.IO;
10
11 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
12
13 namespace Platform.Data.Doublets.Xml {
14     /// <summary>
15     /// <para>
16     /// Represents the xml importer.
17     /// </para>
18     /// <para></para>
19     /// </summary>
20     public class XmlImporter<TLink>
21     {
22         private readonly IXmlStorage<TLink> _storage;
23
24         /// <summary>
25         /// <para>
26         /// Initializes a new <see cref="XmlImporter"/> instance.
27         /// </para>
28         /// <para></para>
29         /// </summary>
30         /// <param name="storage">
31         /// <para>A storage.</para>
32         /// <para></para>
33         /// </param>
34         public XmlImporter(IXmlStorage<TLink> storage) => _storage = storage;
35
36         /// <summary>
37         /// <para>
38         /// Imports the file.
39         /// </para>
40         /// <para></para>
41         /// </summary>
42         /// <param name="file">
43         /// <para>The file.</para>
44         /// <para></para>
45         /// </param>
46         /// <param name="token">
47         /// <para>The token.</para>
48         /// <para></para>
49         /// </param>
50         public Task Import(string file, CancellationToken token)
51         {
52             return Task.Factory.StartNew(() =>
53             {
54                 try
55                 {
56                     var document = _storage.CreateDocument(file);
57
58                     using (var reader = XmlReader.Create(file))
59                     {
60                         Read(reader, token, new ElementContext(document));
61                     }
62                 }
63                 catch (Exception ex)
64                 {
65                     Console.WriteLine(ex.ToStringWithAllInnerExceptions());
66                 }
67             }, token);
68         }
69     }
70
71     private void Read(XmlReader reader, CancellationToken token, ElementContext context)
72     {
73         var parentContexts = new Stack<ElementContext>();
74         var elements = new Stack<string>(); // Path
75         // TODO: If path was loaded previously, skip it.
76         while (reader.Read())
77         {
78             if (token.IsCancellationRequested)
```

```

79     {
80         return;
81     }
82     switch (reader.NodeType)
83     {
84         case XmlNodeType.Element:
85             var elementName = reader.Name;
86             context.IncrementChildNameCount(elementName);
87             elementName =
88                 ↪ $"{elementName}[{context.ChildrenNamesCounts[elementName]}]";
89             if (!reader.IsEmptyElement)
90             {
91                 elements.Push(elementName);
92                 ConsoleHelpers.Debug("{0} starting...", elements.Count <= 20 ?
93                     ↪ ToXPath(elements) : elementName); // XPath
94                 var element = _storage.CreateElement(name: elementName);
95                 parentContexts.Push(context);
96                 _storage.AttachElementToParent(elementToAttach: element, parent:
97                     ↪ context.Parent);
98                 context = new ElementContext(element);
99             }
100             else
101             {
102                 ConsoleHelpers.Debug("{0} finished.", elementName);
103             }
104             break;
105         case XmlNodeType.EndElement:
106             ConsoleHelpers.Debug("{0} finished.", elements.Count <= 20 ?
107                 ↪ ToXPath(elements) : elements.Peek()); // XPath
108             elements.Pop();
109             // Restoring scope
110             context = parentContexts.Pop();
111             if (elements.Count == 1)
112             {
113                 if (context.TotalChildren % 10 == 0)
114                     Console.WriteLine(context.TotalChildren);
115             }
116             break;
117         case XmlNodeType.Text:
118             ConsoleHelpers.Debug("Starting text element...");
119             var content = reader.Value;
120             ConsoleHelpers.Debug("Content: {0}{1}", content.Truncate(50),
121                 ↪ content.Length >= 50 ? "... " : "");
122             var textElement = _storage.CreateTextElement(content: content);
123             _storage.AttachElementToParent(textElement, context.Parent);
124             ConsoleHelpers.Debug("Text element finished.");
125             break;
126     }
127 }
128 }
129 }
130
131 private string ToXPath(Stack<string> path) => string.Join("/", path.Reverse());
132
133 private class ElementContext : XmlElementContext
134 {
135     public readonly TLink Parent;
136     public ElementContext(TLink parent) => Parent = parent;
137 }
138 }

```

1.10 ./csharp/Platform.Data.Doublets.Xml/XmlImporterCLI.cs

```

1  using System;
2  using System.IO;
3  using Platform.IO;
4  using Platform.Data.Doublets.Memory.United.Generic;
5
6  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
7
8  namespace Platform.Data.Doublets.Xml
9  {
10     /// <summary>
11     /// <para>
12     /// Represents the xml importer cli.
13     /// </para>
14     /// <para></para>
15     /// </summary>
16     /// <seealso cref="ICommandLineInterface"/>
17     public class XmlImporterCLI : ICommandLineInterface

```

```

18 {
19     /// <summary>
20     /// <para>
21     /// Runs the args.
22     /// </para>
23     /// <para></para>
24     /// </summary>
25     /// <param name="args">
26     /// <para>The args.</para>
27     /// <para></para>
28     /// </param>
29     public void Run(params string[] args)
30     {
31         var linksFile = ConsoleHelpers.GetOrReadArgument(0, "Links file", args);
32         var file = ConsoleHelpers.GetOrReadArgument(1, "Xml file", args);
33
34         if (!File.Exists(file))
35         {
36             Console.WriteLine("Entered xml file does not exists.");
37         }
38         else
39         {
40             //const long gb32 = 34359738368;
41
42             using (var cancellation = new ConsoleCancellation())
43             using (var memoryAdapter = new UnitedMemoryLinks<uint>(linksFile))
44             //using (var memoryAdapter = new UInt64UnitedMemoryLinks(linksFile, gb32))
45             //using (var links = new UInt64Links(memoryAdapter))
46             {
47                 Console.WriteLine("Press CTRL+C to stop.");
48                 var links =
49                     ↪ memoryAdapter.DecorateWithAutomaticUniquenessAndUsagesResolution();
50                 var indexer = new XmlIndexer<uint>(links);
51                 var indexingImporter = new XmlImporter<uint>(indexer);
52                 indexingImporter.Import(file, cancellation.Token).Wait();
53                 if (cancellation.NotRequested)
54                 {
55                     var cache = indexer.Cache;
56                     //var counter = new TotalSequenceSymbolFrequencyCounter<uint>(links);
57                     //var cache = new LinkFrequenciesCache<uint>(links, counter);
58                     Console.WriteLine("Frequencies cache ready.");
59                     var storage = new DefaultXmlStorage<uint>(links, false, cache);
60                     var importer = new XmlImporter<uint>(storage);
61                     importer.Import(file, cancellation.Token).Wait();
62                 }
63             }
64         }
65     }
66 }

```

1.11 ./csharp/Platform.Data.Doublets.Xml/XmlIndexer.cs

```

1 using System.Collections.Generic;
2 using Platform.Numbers;
3 using Platform.Data.Numbers.Raw;
4 using Platform.Data.Doublets;
5 using Platform.Data.Doublets.Sequences.Frequencies.Cache;
6 using Platform.Data.Doublets.Sequences.Frequencies.Counters;
7 using Platform.Data.Doublets.Sequences.Indexes;
8 using Platform.Data.Doublets.Unicode;
9
10 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
11 namespace Platform.Data.Doublets.Xml
12 {
13     /// <summary>
14     /// <para>
15     /// Represents the xml indexer.
16     /// </para>
17     /// <para></para>
18     /// </summary>
19     /// <seealso cref="IXmlStorage{TLink}" />
20     public class XmlIndexer<TLink> : IXmlStorage<TLink>
21     {
22         private static readonly TLink _zero = default;
23         private static readonly TLink _one = Arithmetic.Increment(_zero);
24         private readonly CachedFrequencyIncrementingSequenceIndex<TLink> _index;
25         private readonly CharToUnicodeSymbolConverter<TLink> _charToUnicodeSymbolConverter;
26         private TLink _unicodeSymbolMarker;
27     }

```

```

28 private readonly TLink _nullConstant;
29
30 /// <summary>
31 /// <para>
32 /// Gets the cache value.
33 /// </para>
34 /// <para></para>
35 /// </summary>
36 public LinkFrequenciesCache<TLink> Cache { get; }
37
38 /// <summary>
39 /// <para>
40 /// Initializes a new <see cref="XmlIndexer"/> instance.
41 /// </para>
42 /// <para></para>
43 /// </summary>
44 /// <param name="links">
45 /// <para>A links.</para>
46 /// <para></para>
47 /// </param>
48 public XmlIndexer(ILinks<TLink> links)
49 {
50     _nullConstant = links.Constants.Null;
51     var totalSequenceSymbolFrequencyCounter = new
52         ↪ TotalSequenceSymbolFrequencyCounter<TLink>(links);
53     Cache = new LinkFrequenciesCache<TLink>(links, totalSequenceSymbolFrequencyCounter);
54     _index = new CachedFrequencyIncrementingSequenceIndex<TLink>(Cache);
55     var addressToRawNumberConverter = new AddressToRawNumberConverter<TLink>();
56     InitConstants(links);
57     _charToUnicodeSymbolConverter = new CharToUnicodeSymbolConverter<TLink>(links,
58         ↪ addressToRawNumberConverter, _unicodeSymbolMarker);
59 }
60
61 private void InitConstants(ILinks<TLink> links)
62 {
63     var markerIndex = _one;
64     var meaningRoot = links.GetOrCreate(markerIndex, markerIndex);
65     _unicodeSymbolMarker = links.GetOrCreate(meaningRoot,
66         ↪ Arithmetic.Increment(markerIndex));
67     _ = links.GetOrCreate(meaningRoot, Arithmetic.Increment(markerIndex));
68     _ = links.GetOrCreate(meaningRoot, Arithmetic.Increment(markerIndex));
69     _ = links.GetOrCreate(meaningRoot, Arithmetic.Increment(markerIndex));
70     _ = links.GetOrCreate(meaningRoot, Arithmetic.Increment(markerIndex));
71 }
72
73 /// <summary>
74 /// <para>
75 /// Attaches the element to parent using the specified element to attach.
76 /// </para>
77 /// <para></para>
78 /// </summary>
79 /// <param name="elementToAttach">
80 /// <para>The element to attach.</para>
81 /// <para></para>
82 /// </param>
83 /// <param name="parent">
84 /// <para>The parent.</para>
85 /// <para></para>
86 /// </param>
87 public void AttachElementToParent(TLink elementToAttach, TLink parent)
88 {
89 }
90
91 /// <summary>
92 /// <para>
93 /// Returns the elements using the specified string.
94 /// </para>
95 /// <para></para>
96 /// </summary>
97 /// <param name="@string">
98 /// <para>The string.</para>
99 /// <para></para>
100 /// </param>
101 /// <returns>
102 /// <para>The elements.</para>
103 /// <para></para>
104 /// </returns>
105 public IList<TLink> ToElements(string @string)

```

```

103 {
104     var elements = new TLink[@string.Length];
105     for (int i = 0; i < @string.Length; i++)
106     {
107         elements[i] = _charToUnicodeSymbolConverter.Convert(@string[i]);
108     }
109     return elements;
110 }
111
112 /// <summary>
113 /// <para>
114 /// Creates the document using the specified name.
115 /// </para>
116 /// <para></para>
117 /// </summary>
118 /// <param name="name">
119 /// <para>The name.</para>
120 /// <para></para>
121 /// </param>
122 /// <returns>
123 /// <para>The null constant.</para>
124 /// <para></para>
125 /// </returns>
126 public TLink CreateDocument(string name)
127 {
128     _index.Add(ToElements(name));
129     return _nullConstant;
130 }
131
132 /// <summary>
133 /// <para>
134 /// Creates the element using the specified name.
135 /// </para>
136 /// <para></para>
137 /// </summary>
138 /// <param name="name">
139 /// <para>The name.</para>
140 /// <para></para>
141 /// </param>
142 /// <returns>
143 /// <para>The null constant.</para>
144 /// <para></para>
145 /// </returns>
146 public TLink CreateElement(string name)
147 {
148     _index.Add(ToElements(name));
149     return _nullConstant;
150 }
151
152 /// <summary>
153 /// <para>
154 /// Creates the text element using the specified content.
155 /// </para>
156 /// <para></para>
157 /// </summary>
158 /// <param name="content">
159 /// <para>The content.</para>
160 /// <para></para>
161 /// </param>
162 /// <returns>
163 /// <para>The null constant.</para>
164 /// <para></para>
165 /// </returns>
166 public TLink CreateTextElement(string content)
167 {
168     _index.Add(ToElements(content));
169     return _nullConstant;
170 }
171
172 /// <summary>
173 /// <para>
174 /// Gets the document using the specified name.
175 /// </para>
176 /// <para></para>
177 /// </summary>
178 /// <param name="name">
179 /// <para>The name.</para>
180 /// <para></para>

```

```

181     /// </param>
182     /// <exception cref="System.NotImplementedException">
183     /// <para></para>
184     /// <para></para>
185     /// </exception>
186     /// <returns>
187     /// <para>The link</para>
188     /// <para></para>
189     /// </returns>
190 public TLink GetDocument(string name)
191 {
192     throw new System.NotImplementedException();
193 }
194
195     /// <summary>
196     /// <para>
197     /// Gets the element using the specified name.
198     /// </para>
199     /// <para></para>
200     /// </summary>
201     /// <param name="name">
202     /// <para>The name.</para>
203     /// <para></para>
204     /// </param>
205     /// <exception cref="System.NotImplementedException">
206     /// <para></para>
207     /// <para></para>
208     /// </exception>
209     /// <returns>
210     /// <para>The link</para>
211     /// <para></para>
212     /// </returns>
213 public TLink GetElement(string name)
214 {
215     throw new System.NotImplementedException();
216 }
217
218     /// <summary>
219     /// <para>
220     /// Gets the text element using the specified content.
221     /// </para>
222     /// <para></para>
223     /// </summary>
224     /// <param name="content">
225     /// <para>The content.</para>
226     /// <para></para>
227     /// </param>
228     /// <exception cref="System.NotImplementedException">
229     /// <para></para>
230     /// <para></para>
231     /// </exception>
232     /// <returns>
233     /// <para>The link</para>
234     /// <para></para>
235     /// </returns>
236 public TLink GetTextElement(string content)
237 {
238     throw new System.NotImplementedException();
239 }
240
241     /// <summary>
242     /// <para>
243     /// Gets the children using the specified parent.
244     /// </para>
245     /// <para></para>
246     /// </summary>
247     /// <param name="parent">
248     /// <para>The parent.</para>
249     /// <para></para>
250     /// </param>
251     /// <exception cref="System.NotImplementedException">
252     /// <para></para>
253     /// <para></para>
254     /// </exception>
255     /// <returns>
256     /// <para>A list of i list t link</para>
257     /// <para></para>
258     /// </returns>

```



```
259     public IList<TLink> GetChildren(TLink parent)
260     {
261         throw new System.NotImplementedException();
262     }
263 }
264 }
```

Index

./csharp/Platform.Data.Doublets.Xml/DefaultXmlStorage.cs, 1
./csharp/Platform.Data.Doublets.Xml/ ICommandLineInterface.cs, 4
./csharp/Platform.Data.Doublets.Xml/IXmlStorage.cs, 4
./csharp/Platform.Data.Doublets.Xml/XmlElementContext.cs, 6
./csharp/Platform.Data.Doublets.Xml/XmlElementCounter.cs, 6
./csharp/Platform.Data.Doublets.Xml/XmlElementCounterCLI.cs, 8
./csharp/Platform.Data.Doublets.Xml/XmlExporter.cs, 9
./csharp/Platform.Data.Doublets.Xml/XmlExporterCLI.cs, 10
./csharp/Platform.Data.Doublets.Xml/XmlImporter.cs, 10
./csharp/Platform.Data.Doublets.Xml/XmlImporterCLI.cs, 12
./csharp/Platform.Data.Doublets.Xml/XmlIndexer.cs, 13