

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             /// <summary>
38             /// <para>
39             /// Determines whether this instance add.
40             /// </para>
41             /// <para></para>
42             /// </summary>
43             /// <param name="sequence">
44             /// <para>The sequence.</para>
45             /// <para></para>
46             /// </param>
47             /// <returns>
48             /// <para>The bool</para>
49             /// <para></para>
50             /// </returns>
51             public bool Add(IList<TLink> sequence) => true;
52
53             /// <summary>
54             /// <para>
55             /// Determines whether this instance might contain.
56             /// </para>
57             /// <para></para>
58             /// </summary>
59             /// <param name="sequence">
60             /// <para>The sequence.</para>
61             /// <para></para>
62             /// </param>
63             /// <returns>
64             /// <para>The bool</para>
65             /// <para></para>
66             /// </returns>
67             public bool MightContain(IList<TLink> sequence) => true;
68         }
69
70         /// <summary>
71         /// <para>
72         /// Initializes a new <see cref="DefaultXmlStorage" /> instance.
73         /// </para>
74         /// <para></para>
75         /// </summary>
76         /// <param name="links">
77         /// <para>A links.</para>
```

```

77     /// <para></para>
78     /// </param>
79     /// <param name="indexSequenceBeforeCreation">
80     /// <para>A index sequence before creation.</para>
81     /// <para></para>
82     /// </param>
83     /// <param name="frequenciesCache">
84     /// <para>A frequencies cache.</para>
85     /// <para></para>
86     /// </param>
87     public DefaultXmlStorage(ILinks<TLink> links, bool indexSequenceBeforeCreation,
88     ↪ LinkFrequenciesCache<TLink> frequenciesCache)
89     {
90         var linkToItsFrequencyNumberConverter = new
91         ↪ FrequenciesCacheBasedLinkToItsFrequencyNumberConverter<TLink>(frequenciesCache);
92         var sequenceToItsLocalElementLevelsConverter = new
93         ↪ SequenceToItsLocalElementLevelsConverter<TLink>(links,
94         ↪ linkToItsFrequencyNumberConverter);
95         var optimalVariantConverter = new OptimalVariantConverter<TLink>(links,
96         ↪ sequenceToItsLocalElementLevelsConverter);
97         InitConstants(links);
98         var charToUnicodeSymbolConverter = new CharToUnicodeSymbolConverter<TLink>(links,
99         ↪ new AddressToRawNumberConverter<TLink>(), _unicodeSymbolMarker);
100        var index = indexSequenceBeforeCreation ? new
101        ↪ CachedFrequencyIncrementingSequenceIndex<TLink>(frequenciesCache) :
102        ↪ (ISequenceIndex<TLink>)new Unindex();
103        _stringToUnicodeSequenceConverter = new
104        ↪ StringToUnicodeSequenceConverter<TLink>(links, charToUnicodeSymbolConverter,
105        ↪ index, optimalVariantConverter, _unicodeSequenceMarker);
106        _links = links;
107    }
108
109     public DefaultXmlStorage(ILinks<TLink> links, bool indexSequenceBeforeCreation = false) :
110     this(links, indexSequenceBeforeCreation,
111         new LinkFrequenciesCache<TLink>(links,
112         new TotalSequenceSymbolFrequencyCounter<TLink>(links))) { }
113
114     private void InitConstants(ILinks<TLink> links)
115     {
116         var markerIndex = _one;
117         var meaningRoot = links.GetOrCreate(markerIndex, markerIndex);
118         _unicodeSymbolMarker = links.GetOrCreate(meaningRoot, Arithmetic.Increment(ref
119         ↪ markerIndex));
120         _unicodeSequenceMarker = links.GetOrCreate(meaningRoot, Arithmetic.Increment(ref
121         ↪ markerIndex));
122         _elementMarker = links.GetOrCreate(meaningRoot, Arithmetic.Increment(ref
123         ↪ markerIndex));
124         _textElementMarker = links.GetOrCreate(meaningRoot, Arithmetic.Increment(ref
125         ↪ markerIndex));
126         _documentMarker = links.GetOrCreate(meaningRoot, Arithmetic.Increment(ref
127         ↪ markerIndex));
128     }
129
130     /// <summary>
131     /// <para>
132     /// Creates the document using the specified name.
133     /// </para>
134     /// <para></para>
135     /// </summary>
136     /// <param name="name">
137     /// <para>The name.</para>
138     /// <para></para>
139     /// </param>
140     /// <returns>
141     /// <para>The link</para>
142     /// <para></para>
143     /// </returns>
144     public TLink CreateDocument(string name) => Create(_documentMarker, name);
145
146     /// <summary>
147     /// <para>
148     /// Creates the element using the specified name.
149     /// </para>
150     /// <para></para>
151     /// </summary>
152     /// <param name="name">
153     /// <para>The name.</para>
154     /// <para></para>

```

```

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

```

```

215     /// </para>
216     /// <para></para>
217     /// </summary>
218     /// <param name="name">
219     /// <para>The name.</para>
220     /// <para></para>
221     /// </param>
222     /// <returns>
223     /// <para>The link</para>
224     /// <para></para>
225     /// </returns>
226     public TLink GetElement(string name) => Get(_elementMarker, name);
227
228     private TLink Get(TLink marker, string content) => _links.SearchOrDefault(marker,
229         ↪ _stringToUnicodeSequenceConverter.Convert(content));
230
231     /// <summary>
232     /// <para>
233     /// Gets the children using the specified parent.
234     /// </para>
235     /// <para></para>
236     /// </summary>
237     /// <param name="parent">
238     /// <para>The parent.</para>
239     /// <para></para>
240     /// </param>
241     /// <returns>
242     /// <para>The childrens.</para>
243     /// <para></para>
244     /// </returns>
245     public IList<TLink> GetChildren(TLink parent) {
246         List<TLink> childrens = new List<TLink>();
247         _links.Each((link) => {
248             childrens.Add(_links.GetTarget(link));
249             return this._links.Constants.Continue;
250         }, new Link<TLink>(_links.Constants.Any, parent, _links.Constants.Any));
251         return childrens;
252     }
253 }

```

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      /// <summary>
31      /// <para>
32      /// Creates the element using the specified name.
33      /// </para>
34      /// <para></para>
35      /// </summary>
36      /// <param name="name">
37      /// <para>The name.</para>
38      /// <para></para>
39      /// </param>
40      /// <returns>
41      /// <para>The link</para>
42      /// <para></para>
43      /// </returns>
44      TLink CreateElement(string name);
45      /// <summary>
46      /// <para>
47      /// Creates the text element using the specified content.
48      /// </para>
49      /// <para></para>
50      /// </summary>
51      /// <param name="content">
52      /// <para>The content.</para>
53      /// <para></para>
54      /// </param>
55      /// <returns>
56      /// <para>The link</para>
57      /// <para></para>
58      /// </returns>
59      TLink CreateTextElement(string content);
60      /// <summary>
61      /// <para>
62      /// Gets the document using the specified name.
63      /// </para>
64      /// <para></para>
65      /// </summary>
66      /// <param name="name">
67      /// <para>The name.</para>
68      /// <para></para>
69      /// </param>
70      /// <returns>
71      /// <para>The link</para>
72      /// <para></para>
73      /// </returns>
74      TLink GetDocument(string name);
75      /// <summary>
76      /// <para>
77      /// Gets the element using the specified name.
78      /// </para>
79      /// <para></para>
80      /// </summary>
81      /// <param name="name">
82      /// <para>The name.</para>
83      /// <para></para>
84      /// </param>
85      /// <returns>
86      /// <para>The link</para>
87      /// <para></para>

```

```

88     /// </returns>
89     TLink GetElement(string name);
90     /// <summary>
91     /// <para>
92     /// Gets the text element using the specified content.
93     /// </para>
94     /// <para></para>
95     /// </summary>
96     /// <param name="content">
97     /// <para>The content.</para>
98     /// <para></para>
99     /// </param>
100    /// <returns>
101    /// <para>The link</para>
102    /// <para></para>
103    /// </returns>
104    TLink GetTextElement(string content);
105    /// <summary>
106    /// <para>
107    /// Gets the children using the specified parent.
108    /// </para>
109    /// <para></para>
110    /// </summary>
111    /// <param name="parent">
112    /// <para>The parent.</para>
113    /// <para></para>
114    /// </param>
115    /// <returns>
116    /// <para>A list of i list t link</para>
117    /// <para></para>
118    /// </returns>
119    IList<TLink> GetChildren(TLink parent);
120    /// <summary>
121    /// <para>
122    /// Attaches the element to parent using the specified element to attach.
123    /// </para>
124    /// <para></para>
125    /// </summary>
126    /// <param name="elementToAttach">
127    /// <para>The element to attach.</para>
128    /// <para></para>
129    /// </param>
130    /// <param name="parent">
131    /// <para>The parent.</para>
132    /// <para></para>
133    /// </param>
134    void AttachElementToParent(TLink elementToAttach, TLink parent);
135 }
136 }

```

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         {

```

```

70     var rootContext = (RootElementContext)context;
71     var parentContexts = new Stack<XmlElementContext>();
72     var elements = new Stack<string>(); // Path
73     // TODO: If path was loaded previously, skip it.
74     while (reader.Read())
75     {
76         if (token.IsCancellationRequested)
77         {
78             return;
79         }
80         switch (reader.NodeType)
81         {
82             case XmlNodeType.Element:
83                 var elementName = reader.Name;
84                 context.IncrementChildNameCount(elementName);
85                 elementName =
86                     ↳ $"{elementName}[{context.ChildrenNamesCounts[elementName]}]";
87                 if (!reader.IsEmptyElement)
88                 {
89                     elements.Push(elementName);
90                     ConsoleHelpers.Debug("{0} starting...", elements.Count <= 20 ?
91                         ↳ ToXPath(elements) : elementName); // XPath
92                     parentContexts.Push(context);
93                     context = new XmlElementContext();
94                 }
95                 else
96                 {
97                     ConsoleHelpers.Debug("{0} finished.", elementName);
98                 }
99                 break;
100             case XmlNodeType.EndElement:
101                 ConsoleHelpers.Debug("{0} finished.", elements.Count <= 20 ?
102                     ↳ ToXPath(elements) : elements.Peek()); // XPath
103                 var topElement = elements.Pop();
104                 // Restoring scope
105                 context = parentContexts.Pop();
106                 if (topElement.StartsWith(elementNameToCount))
107                 {
108                     rootContext.TotalElements++;
109                     // TODO: Check for 0x00 part/symbol at 198102797 line and 13
110                     ↳ position.
111                     //if (rootContext.TotalPages > 3490000)
112                     //    selfCancel = true;
113                     if (context.ChildrenNamesCounts[elementNameToCount] % 10000 == 0)
114                     {
115                         Console.WriteLine(topElement);
116                     }
117                 }
118                 break;
119             case XmlNodeType.Text:
120                 ConsoleHelpers.Debug("Starting text element...");
121                 var content = reader.Value;
122                 rootContext.TotalContentsLength += (ulong)content.Length;
123                 ConsoleHelpers.Debug($"{Content length is: {content.Length}");
124                 ConsoleHelpers.Debug("Text element finished.");
125                 break;
126         }
127     }
128 }
129
130 private string ToXPath(Stack<string> path) => string.Join("/", path.Reverse());
131
132 private class RootElementContext : XmlElementContext
133 {
134     /// <summary>
135     /// <para>
136     /// The total elements.
137     /// </para>
138     /// <para></para>
139     /// </summary>
140     public ulong TotalElements;
141     /// <summary>
142     /// <para>
143     /// The total contents length.
144     /// </para>
145     /// <para></para>
146     /// </summary>

```



```

145         public ulong TotalContentsLength;
146     }
147 }
148 }

```

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         {
19             Console.WriteLine("Entered xml file does already exists.");
20         }
21         else
22         {
23             using (var cancellation = new ConsoleCancellation())
24             {
25                 using (var memoryAdapter = new UnitedMemoryLinks<uint>(linksFile))
26                 {
27                     {
28                         Console.WriteLine("Press CTRL+C to stop.");
29                         var links = memoryAdapter.DecorateWithAutomaticUniquenessAndUsagesRe
30                             ↪ solution();
31                         if (cancellation.NotRequested)
32                         {
33                             var storage = new DefaultXmlStorage<uint>(links);
34                             var exporter = new XmlExporter<uint>(storage);
35                             exporter.Export(linksFile, exportFile,
36                                 ↪ cancellation.Token).Wait();
37                         }
38                     }
39                 }
40             }
41         }
42     }

```

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  private void Read(XmlReader reader, CancellationToken token, ElementContext context)
71  {
72      var parentContexts = new Stack<ElementContext>();
73      var elements = new Stack<string>(); // Path
74      // TODO: If path was loaded previously, skip it.
75      while (reader.Read())
76      {
77          if (token.IsCancellationRequested)
78          {
79              return;
80          }
81          switch (reader.NodeType)
82          {
83              case XmlNodeType.Element:
84                  var elementName = reader.Name;
85                  context.IncrementChildNameCount(elementName);
86                  elementName =
87                      ↪ $"{elementName}[{context.ChildrenNamesCounts[elementName]}]";
88                  if (!reader.IsEmptyElement)
89                  {
90                      elements.Push(elementName);
91                      ConsoleHelpers.Debug("{0} starting...", elements.Count <= 20 ?
92                          ↪ ToXPath(elements) : elementName); // XPath
93                      var element = _storage.CreateElement(name: elementName);
94                      parentContexts.Push(context);
95                      _storage.AttachElementToParent(elementToAttach: element, parent:
96                          ↪ context.Parent);
97                      context = new ElementContext(element);
98                  }
99                  else
100                  {
101                      ConsoleHelpers.Debug("{0} finished.", elementName);
102                      break;
103                  }
104              case XmlNodeType.EndElement:
105                  ConsoleHelpers.Debug("{0} finished.", elements.Count <= 20 ?
106                      ↪ ToXPath(elements) : elements.Peek()); // XPath
107                  elements.Pop();
108                  // Restoring scope
109                  context = parentContexts.Pop();
110                  if (elements.Count == 1)
111                  {
112                      if (context.TotalChildren % 10 == 0)
113                          Console.WriteLine(context.TotalChildren);
114                      break;
115                  }
116              case XmlNodeType.Text:
117                  ConsoleHelpers.Debug("Starting text element...");
118                  var content = reader.Value;

```

```

116         ConsoleHelpers.Debug("Content: {0}{1}", content.Truncate(50),
117                               ↳ content.Length >= 50 ? "... " : "");
118         var textElement = _storage.CreateTextElement(content: content);
119         _storage.AttachElementToParent(textElement, context.Parent);
120         ConsoleHelpers.Debug("Text element finished.");
121         break;
122     }
123 }
124
125 private string ToXPath(Stack<string> path) => string.Join("/", path.Reverse());
126
127 private class ElementContext : XmlElementContext
128 {
129     public readonly TLink Parent;
130     public ElementContext(TLink parent) => Parent = parent;
131 }
132 }
133 }
134 }

```

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);

```

```

57         Console.WriteLine("Frequencies cache ready.");
58         var storage = new DefaultXmlStorage<uint>(links, false, cache);
59         var importer = new XmlImporter<uint>(storage);
60         importer.Import(file, cancellation.Token).Wait();
61     }
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
12 namespace Platform.Data.Doublets.Xml
13 {
14     /// <summary>
15     /// <para>
16     /// Represents the xml indexer.
17     /// </para>
18     /// <para></para>
19     /// </summary>
20     /// <seealso cref="IXmlStorage{TLink}" />
21     public class XmlIndexer<TLink> : IXmlStorage<TLink>
22     {
23         private static readonly TLink _zero = default;
24         private static readonly TLink _one = Arithmetic.Increment(_zero);
25         private readonly CachedFrequencyIncrementingSequenceIndex<TLink> _index;
26         private readonly CharToUnicodeSymbolConverter<TLink> _charToUnicodeSymbolConverter;
27         private TLink _unicodeSymbolMarker;
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         }
69     }
70 }

```

```

65         _ = links.GetOrCreate(meaningRoot, Arithmetic.Increment(markerIndex));
66         _ = links.GetOrCreate(meaningRoot, Arithmetic.Increment(markerIndex));
67         _ = links.GetOrCreate(meaningRoot, Arithmetic.Increment(markerIndex));
68     }
69
70     /// <summary>
71     /// <para>
72     /// Attaches the element to parent using the specified element to attach.
73     /// </para>
74     /// <para></para>
75     /// </summary>
76     /// <param name="elementToAttach">
77     /// <para>The element to attach.</para>
78     /// <para></para>
79     /// </param>
80     /// <param name="parent">
81     /// <para>The parent.</para>
82     /// <para></para>
83     /// </param>
84     public void AttachElementToParent(TLink elementToAttach, TLink parent)
85     {
86     }
87
88     /// <summary>
89     /// <para>
90     /// Returns the elements using the specified string.
91     /// </para>
92     /// <para></para>
93     /// </summary>
94     /// <param name="@string">
95     /// <para>The string.</para>
96     /// <para></para>
97     /// </param>
98     /// <returns>
99     /// <para>The elements.</para>
100    /// <para></para>
101    /// </returns>
102    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, 7
./csharp/Platform.Data.Doublets.Xml/XmlElementCounterCLI.cs, 9
./csharp/Platform.Data.Doublets.Xml/XmlExporter.cs, 9
./csharp/Platform.Data.Doublets.Xml/XmlExporterCLI.cs, 10
./csharp/Platform.Data.Doublets.Xml/XmlImporter.cs, 11
./csharp/Platform.Data.Doublets.Xml/XmlImporterCLI.cs, 13
./csharp/Platform.Data.Doublets.Xml/XmlIndexer.cs, 14