

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

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

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

```

62     public TLink GetElement(string name) => Get(_elementMarker, name);
63
64     public TLink CreateTextElement(string content) => Create(_textElementMarker, content);
65
66     public TLink GetTextElement(string content) => Get(_textElementMarker, content);
67
68     private TLink Create(TLink marker, string content) => _links.GetOrCreate(marker,
69         ↪ _stringToUnicodeSequenceConverter.Convert(content));
70
71     private TLink Get(TLink marker, string content) => _links.SearchOrDefault(marker,
72         ↪ _stringToUnicodeSequenceConverter.Convert(content));
73
74     public void AttachElementToParent(TLink elementToAttach, TLink parent) =>
75         ↪ _links.GetOrCreate(parent, elementToAttach);
76 }
77 }

```

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      public interface ICommandLineInterface
6      {
7          void Run(params string[] args);
8      }
9  }

```

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  namespace Platform.Data.Doublets.Xml
4  {
5      public interface IXmlStorage<TLink>
6      {
7          TLink CreateDocument(string name);
8          TLink CreateElement(string name);
9          TLink CreateTextElement(string content);
10         TLink GetDocument(string name);
11         TLink GetElement(string name);
12         TLink GetTextElement(string content);
13         void AttachElementToParent(TLink elementToAttach, TLink parent);
14     }
15 }

```

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      internal class XmlElementContext
8      {
9          public readonly Dictionary<string, int> ChildrenNamesCounts;
10         public int TotalChildren;
11
12         public XmlElementContext() => ChildrenNamesCounts = new Dictionary<string, int>();
13
14         public void IncrementChildNameCount(string name)
15         {
16             if (ChildrenNamesCounts.TryGetValue(name, out int count))
17             {
18                 ChildrenNamesCounts[name] = count + 1;
19             }
20             else
21             {
22                 ChildrenNamesCounts[name] = 0;
23             }
24             TotalChildren++;
25         }
26     }
27 }

```

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     public class XmlElementCounter
15     {
16         public XmlElementCounter() { }
17
18         public Task Count(string file, string elementName, CancellationToken token)
19         {
20             return Task.Factory.StartNew(() =>
21             {
22                 try
23                 {
24                     var context = new RootElementContext();
25                     using (var reader = XmlReader.Create(file))
26                     {
27                         Count(reader, elementName, token, context);
28                     }
29                     Console.WriteLine($"Total elements with specified name:
30                                     ↪ {context.TotalElements}, total content length:
31                                     ↪ {context.TotalContentsLength}.");
32                 }
33                 catch (Exception ex)
34                 {
35                     Console.WriteLine(ex.ToStringWithAllInnerExceptions());
36                 }
37             }, token);
38
39     private void Count(XmlReader reader, string elementNameToCount, CancellationToken token,
40 ↪ XmlElementContext context)
41     {
42         var rootContext = (RootElementContext)context;
43         var parentContexts = new Stack<XmlElementContext>();
44         var elements = new Stack<string>(); // Path
45         // TODO: If path was loaded previously, skip it.
46         while (reader.Read())
47         {
48             if (token.IsCancellationRequested)
49             {
50                 return;
51             }
52             switch (reader.NodeType)
53             {
54                 case XmlNodeType.Element:
55                     var elementName = reader.Name;
56                     context.IncrementChildNameCount(elementName);
57                     elementName =
58                         ↪ $"[{elementName}]{context.ChildrenNamesCounts[elementName]}";
59                     if (!reader.IsEmptyElement)
60                     {
61                         elements.Push(elementName);
62                         ConsoleHelpers.Debug("{0} starting...", elements.Count <= 20 ?
63                                             ↪ ToXPath(elements) : elementName); // XPath
64                         parentContexts.Push(context);
65                         context = new XmlElementContext();
66                     }
67                     else
68                     {
69                         ConsoleHelpers.Debug("{0} finished.", elementName);
70                     }
71                     break;
72                 case XmlNodeType.EndElement:
73                     ConsoleHelpers.Debug("{0} finished.", elements.Count <= 20 ?
74                                             ↪ ToXPath(elements) : elements.Peek()); // XPath
75                     var topElement = elements.Pop();
76                     // Restoring scope
77                     context = parentContexts.Pop();
78                     if (topElement.StartsWith(elementNameToCount))
79                     {
80                         rootContext.TotalElements++;
81                     }
82                 }
83             }
84         }
85     }
86 }

```

```

77         // TODO: Check for 0x00 part/symbol at 198102797 line and 13
78         → position.
79         //if (rootContext.TotalPages > 3490000)
80         //    selfCancel = true;
81         if (context.ChildrenNamesCounts[elementNameToCount] % 10000 == 0)
82         {
83             Console.WriteLine(topElement);
84         }
85         break;
86
87         case XmlNodeType.Text:
88             ConsoleHelpers.Debug("Starting text element...");
89             var content = reader.Value;
90             rootContext.TotalContentsLength += (ulong)content.Length;
91             ConsoleHelpers.Debug($"Content length is: {content.Length}");
92             ConsoleHelpers.Debug("Text element finished.");
93             break;
94         }
95     }
96 }
97
98 private string ToXPath(Stack<string> path) => string.Join("/", path.Reverse());
99
100 private class RootElementContext : XmlElementContext
101 {
102     public ulong TotalElements;
103     public ulong TotalContentsLength;
104 }
105 }
106 }

```

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      public class XmlElementCounterCLI : ICommandLineInterface
10     {
11         public void Run(params string[] args)
12         {
13             var file = ConsoleHelpers.GetOrReadArgument(0, "Xml file", args);
14             var elementName = ConsoleHelpers.GetOrReadArgument(1, "Element name to count", args);
15             if (!File.Exists(file))
16             {
17                 Console.WriteLine("Entered xml file does not exists.");
18             }
19             else if (string.IsNullOrEmpty(elementName))
20             {
21                 Console.WriteLine("Entered element name is empty.");
22             }
23             else
24             {
25                 using (var cancellation = new ConsoleCancellation())
26                 {
27                     Console.WriteLine("Press CTRL+C to stop.");
28                     new XmlElementCounter().Count(file, elementName, cancellation.Token).Wait();
29                 }
30             }
31         }
32     }
33 }

```

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  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 {
15     class XmlExporter<TLink>
16     {
17         private readonly IXmlStorage<TLink> _storage;
18
19         public XmlExporter(IXmlStorage<TLink> storage) => _storage = storage;
20
21         public Task Export(string file, CancellationToken token) { return default; }
22
23         private string ToXPath(Stack<string> path) => string.Join("/", path.Reverse());
24
25         private class ElementContext : XmlElementContext
26         {
27             public readonly TLink Parent;
28
29             public ElementContext(TLink parent) => Parent = parent;
30         }
31     }
32 }
33 }

```

1.8 ./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     public class XmlImporter<TLink>
15     {
16         private readonly IXmlStorage<TLink> _storage;
17
18         public XmlImporter(IXmlStorage<TLink> storage) => _storage = storage;
19
20         public Task Import(string file, CancellationToken token)
21         {
22             return Task.Factory.StartNew(() =>
23             {
24                 try
25                 {
26                     var document = _storage.CreateDocument(file);
27
28                     using (var reader = XmlReader.Create(file))
29                     {
30                         Read(reader, token, new ElementContext(document));
31                     }
32                 }
33                 catch (Exception ex)
34                 {
35                     Console.WriteLine(ex.ToStringWithAllInnerExceptions());
36                 }
37             }, token);
38         }
39
40         private void Read(XmlReader reader, CancellationToken token, ElementContext context)
41         {
42             var parentContexts = new Stack<ElementContext>();
43             var elements = new Stack<string>(); // Path
44             // TODO: If path was loaded previously, skip it.
45             while (reader.Read())
46             {
47                 if (token.IsCancellationRequested)
48                 {
49                     return;
50                 }
51                 switch (reader.NodeType)
52                 {
53                     case XmlNodeType.Element:
54                         var elementName = reader.Name;
55                         context.IncrementChildNameCount(elementName);
56                         elementName =
57                             ↪ $"{elementName}[{context.ChildrenNamesCounts[elementName]}]";

```

```

58         if (!reader.IsEmptyElement)
59         {
60             elements.Push(elementName);
61             ConsoleHelpers.Debug("{0} starting...", elements.Count <= 20 ?
                ↳ ToXPath(elements) : elementName); // XPath
62             var element = _storage.CreateElement(name: elementName);
63             parentContexts.Push(context);
64             _storage.AttachElementToParent(elementToAttach: element, parent:
                ↳ context.Parent);
65             context = new ElementContext(element);
66         }
67         else
68         {
69             ConsoleHelpers.Debug("{0} finished.", elementName);
70         }
71         break;
72     case XmlNodeType.EndElement:
73         ConsoleHelpers.Debug("{0} finished.", elements.Count <= 20 ?
                ↳ ToXPath(elements) : elements.Peek()); // XPath
74         elements.Pop();
75         // Restoring scope
76         context = parentContexts.Pop();
77         if (elements.Count == 1)
78         {
79             if (context.TotalChildren % 10 == 0)
80                 Console.WriteLine(context.TotalChildren);
81         }
82         break;
83     case XmlNodeType.Text:
84         ConsoleHelpers.Debug("Starting text element...");
85         var content = reader.Value;
86         ConsoleHelpers.Debug("Content: {0}{1}", content.Truncate(50),
                ↳ content.Length >= 50 ? "... " : "");
87         var textElement = _storage.CreateTextElement(content: content);
88         _storage.AttachElementToParent(textElement, context.Parent);
89         ConsoleHelpers.Debug("Text element finished.");
90         break;
91     }
92 }
93 }
94
95 private string ToXPath(Stack<string> path) => string.Join("/", path.Reverse());
96
97 private class ElementContext : XmlElementContext
98 {
99     public readonly TLink Parent;
100
101     public ElementContext(TLink parent) => Parent = parent;
102 }
103 }
104 }

```

1.9 ./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     public class XmlImporterCLI : ICommandLineInterface
11     {
12         public void Run(params string[] args)
13         {
14             var linksFile = ConsoleHelpers.GetOrReadArgument(0, "Links file", args);
15             var file = ConsoleHelpers.GetOrReadArgument(1, "Xml file", args);
16
17             if (!File.Exists(file))
18             {
19                 Console.WriteLine("Entered xml file does not exists.");
20             }
21             else
22             {
23                 //const long gb32 = 34359738368;
24
25                 using (var cancellation = new ConsoleCancellation())
26                 using (var memoryAdapter = new UnitedMemoryLinks<uint>(linksFile))

```

```

27 //using (var memoryAdapter = new UInt64UnitedMemoryLinks(linksFile, gb32))
28 //using (var links = new UInt64Links(memoryAdapter))
29 {
30     Console.WriteLine("Press CTRL+C to stop.");
31     var links =
32         ↪ memoryAdapter.DecorateWithAutomaticUniquenessAndUsagesResolution();
33     var indexer = new XmlIndexer<uint>(links);
34     var indexingImporter = new XmlImporter<uint>(indexer);
35     indexingImporter.Import(file, cancellation.Token).Wait();
36     if (cancellation.NotRequested)
37     {
38         var cache = indexer.Cache;
39         //var counter = new TotalSequenceSymbolFrequencyCounter<uint>(links);
40         //var cache = new LinkFrequenciesCache<uint>(links, counter);
41         Console.WriteLine("Frequencies cache ready.");
42         var storage = new DefaultXmlStorage<uint>(links, false, cache);
43         var importer = new XmlImporter<uint>(storage);
44         importer.Import(file, cancellation.Token).Wait();
45     }
46 }
47 }
48 }
49 }

```

1.10 ./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     public class XmlIndexer<TLink> : IXmlStorage<TLink>
15     {
16         private static readonly TLink _zero = default;
17         private static readonly TLink _one = Arithmetic.Increment(_zero);
18
19         private readonly CachedFrequencyIncrementingSequenceIndex<TLink> _index;
20         private readonly CharToUnicodeSymbolConverter<TLink> _charToUnicodeSymbolConverter;
21         private TLink _unicodeSymbolMarker;
22         private readonly TLink _nullConstant;
23
24         public LinkFrequenciesCache<TLink> Cache { get; }
25
26         public XmlIndexer(ILinks<TLink> links)
27         {
28             _nullConstant = links.Constants.Null;
29             var totalSequenceSymbolFrequencyCounter = new
30                 ↪ TotalSequenceSymbolFrequencyCounter<TLink>(links);
31             Cache = new LinkFrequenciesCache<TLink>(links, totalSequenceSymbolFrequencyCounter);
32             _index = new CachedFrequencyIncrementingSequenceIndex<TLink>(Cache);
33             var addressToRawNumberConverter = new AddressToRawNumberConverter<TLink>();
34             InitConstants(links);
35             _charToUnicodeSymbolConverter = new CharToUnicodeSymbolConverter<TLink>(links,
36                 ↪ addressToRawNumberConverter, _unicodeSymbolMarker);
37         }
38
39         private void InitConstants(ILinks<TLink> links)
40         {
41             var markerIndex = _one;
42             var meaningRoot = links.GetOrCreate(markerIndex, markerIndex);
43             _unicodeSymbolMarker = links.GetOrCreate(meaningRoot,
44                 ↪ Arithmetic.Increment(markerIndex));
45             _ = links.GetOrCreate(meaningRoot, Arithmetic.Increment(markerIndex));
46             _ = links.GetOrCreate(meaningRoot, Arithmetic.Increment(markerIndex));
47             _ = links.GetOrCreate(meaningRoot, Arithmetic.Increment(markerIndex));
48             _ = links.GetOrCreate(meaningRoot, Arithmetic.Increment(markerIndex));
49         }
50
51         public void AttachElementToParent(TLink elementToAttach, TLink parent)
52         {
53         }
54     }
55 }

```

```

52 public IList<TLink> ToElements(string @string)
53 {
54     var elements = new TLink[@string.Length];
55     for (int i = 0; i < @string.Length; i++)
56     {
57         elements[i] = _charToUnicodeSymbolConverter.Convert(@string[i]);
58     }
59     return elements;
60 }
61
62 public TLink CreateDocument(string name)
63 {
64     _index.Add(ToElements(name));
65     return _nullConstant;
66 }
67
68 public TLink CreateElement(string name)
69 {
70     _index.Add(ToElements(name));
71     return _nullConstant;
72 }
73
74 public TLink CreateTextElement(string content)
75 {
76     _index.Add(ToElements(content));
77     return _nullConstant;
78 }
79 }
80 }

```


Index

./csharp/Platform.Data.Doublets.Xml/DefaultXmlStorage.cs, 1
./csharp/Platform.Data.Doublets.Xml/ ICommandLineInterface.cs, 2
./csharp/Platform.Data.Doublets.Xml/IXmlStorage.cs, 2
./csharp/Platform.Data.Doublets.Xml/XmlElementContext.cs, 2
./csharp/Platform.Data.Doublets.Xml/XmlElementCounter.cs, 2
./csharp/Platform.Data.Doublets.Xml/XmlElementCounterCLI.cs, 4
./csharp/Platform.Data.Doublets.Xml/XmlExporter.cs, 4
./csharp/Platform.Data.Doublets.Xml/XmlImporter.cs, 5
./csharp/Platform.Data.Doublets.Xml/XmlImporterCLI.cs, 6
./csharp/Platform.Data.Doublets.Xml/XmlIndexer.cs, 7