LinksPlatform's Platform.Data.Doublets Class Library

./Converters/AddressToUnaryNumberConverter.cs ./Converters/LinkToItsFrequencyNumberConveter.cs ./Converters/PowerOf2ToUnaryNumberConverter.cs ./Converters/UnaryNumberToAddressAddOperationConverter.cs ./Converters/UnaryNumberToAddressOrOperationConverter.cs ./Decorators/LinksCascadeDependenciesResolver.cs ./Decorators/LinksCascadeUniquenessAndDependenciesResolver.cs ./Decorators/LinksDecoratorBase.cs ./Decorators/LinksDependenciesValidator.cs ./Decorators/LinksDisposableDecoratorBase.cs ./Decorators/LinksInnerReferenceValidator.cs ./Decorators/LinksNonExistentReferencesCreator.cs ./Decorators/LinksNullToSelfReferenceResolver.cs ./Decorators/LinksSelfReferenceResolver.cs ./Decorators/LinksUniquenessResolver.cs ./Decorators/LinksUniquenessValidator.cs ./Decorators/NonNullContentsLinkDeletionResolver.cs ./Decorators/UInt64Links.cs ./Decorators/UniLinks.cs ./DoubletComparer.cs ./Doublet.cs ./Hybrid.cs ./ILinks.cs ./ILinksExtensions.cs ./Incrementers/FrequencyIncrementer.cs ./Incrementers/LinkFrequencyIncrementer.cs ./Incrementers/UnaryNumberIncrementer.cs ./ISynchronizedLinks.cs ./Link.cs ./LinkExtensions.cs ./LinksOperatorBase.cs ./obj/Debug/netstandard2.0/Platform.Data.Doublets.AssemblyInfo.cs ./PropertyOperators/DefaultLinkPropertyOperator.cs ./PropertyOperators/FrequencyPropertyOperator.cs ./ResizableDirectMemory/ResizableDirectMemoryLinks.cs ./ResizableDirectMemory/ResizableDirectMemoryLinks.ListMethods.cs ./ResizableDirectMemory/ResizableDirectMemoryLinks.TreeMethods.cs ./ResizableDirectMemory/UInt64ResizableDirectMemoryLinks.cs ./ResizableDirectMemory/UInt64ResizableDirectMemoryLinks.ListMethods.cs ./ResizableDirectMemory/UInt64ResizableDirectMemoryLinks.TreeMethods.cs ./Sequences/Converters/BalancedVariantConverter.cs ./Sequences/Converters/CompressingConverter.cs ./Sequences/Converters/LinksListToSequenceConverterBase.cs ./Sequences/Converters/OptimalVariantConverter.cs ./Sequences/Converters/SequenceToItsLocalElementLevelsConverter.cs ./Sequences/CreteriaMatchers/DefaultSequenceElementCreteriaMatcher.cs ./Sequences/CreteriaMatchers/MarkedSequenceCreteriaMatcher.cs ./Sequences/DefaultSequenceAppender.cs ./Sequences/DuplicateSegmentsCounter.cs ./Sequences/DuplicateSegmentsProvider.cs ./Sequences/Frequencies/Cache/FrequenciesCacheBasedLinkFrequencyIncrementer.cs ./Sequences/Frequencies/Cache/FrequenciesCacheBasedLinkToItsFrequencyNumberConverter.cs ./Sequences/Frequencies/Cache/LinkFrequenciesCache.cs ./Sequences/Frequencies/Cache/LinkFrequency.cs ./Sequences/Frequencies/Counters/MarkedSequenceSymbolFrequencyOneOffCounter.cs ./Sequences/Frequencies/Counters/SequenceSymbolFrequencyOneOffCounter.cs ./Sequences/Frequencies/Counters/TotalMarkedSequenceSymbolFrequencyCounter.cs ./Sequences/Frequencies/Counters/TotalMarkedSequenceSymbolFrequencyOneOffCounter.cs ./Sequences/Frequencies/Counters/TotalSequenceSymbolFrequencyCounter.cs ./Sequences/Frequencies/Counters/TotalSequenceSymbolFrequencyOneOffCounter.cs ./Sequences/HeightProviders/CachedSequenceHeightProvider.cs ./Sequences/HeightProviders/DefaultSequenceRightHeightProvider.cs ./Sequences/HeightProviders/ISequenceHeightProvider.cs ./Sequences/Sequences.cs ./Sequences/Sequences.Experiments.cs ./Sequences/Sequences.Experiments.ReadSequence.cs ./Sequences/SequencesExtensions.cs ./Sequences/SequencesIndexer.cs ./Sequences/SequencesOptions.cs ./Sequences/UnicodeMap.cs ./Sequences/Walkers/LeftSequenceWalker.cs ./Sequences/Walkers/RightSequenceWalker.cs ./Sequences/Walkers/SequenceWalkerBase.cs ./Stacks/Stack.cs ./Stacks/StackExtensions.cs ./SynchronizedLinks.cs ./UInt64Link.cs ./UInt64LinkExtensions.cs ./UInt64LinksExtensions.cs ./UInt64LinksTransactionsLayer.cs

./UInt64LinksTransactionsLayer.cs

```
using System;
using System.Linq;
using System.Collections.Generic;
using System.IO;
using System.Runtime.CompilerServices;
using System.Threading;
using System.Threading.Tasks;
using Platform.Disposables;
using Platform.Timestamps;
using Platform.Unsafe;
using Platform.IO;
using Platform.Data.Doublets.Decorators;

namespace Platform.Data.Doublets
{
    public class UInt64LinksTransactionsLayer : LinksDisposableDecoratorBase<ulong>
    ↪   //-V3073
    {
        /// <remarks>
        /// Альтернативные варианты хранения трансформации (элемента транзакции):
        ///
        /// private enum TransitionType
        /// {
        ///     Creation,
        ///     UpdateOf,
        ///     UpdateTo,
        ///     Deletion
        /// }
        ///
        /// private struct Transition
        /// {
        ///     public ulong TransactionId;
        ///     public UniqueTimestamp Timestamp;
        ///     public TransactionItemType Type;
        ///     public Link Source;
        ///     public Link Linker;
        ///     public Link Target;
        /// }
        ///
        /// Или
        ///
        /// public struct TransitionHeader
        /// {
        ///     public ulong TransactionIdCombined;
        ///     public ulong TimestampCombined;
        ///
        ///     public ulong TransactionId
        ///     {
        ///         get
        ///         {
        ///             return (ulong) mask & TransactionIdCombined;
        ///         }
        ///     }
        ///
        ///     public UniqueTimestamp Timestamp
        ///     {
        ///         get
        ///         {
        ///             return (UniqueTimestamp)mask & TransactionIdCombined;
        ///         }
        ///     }
        ///
        ///     public TransactionItemType Type
        ///     {
        ///         get
        ///         {
        ///             // Использовать по одному биту из TransactionId и Timestamp,
        ///             // для значения в 2 бита, которое представляет тип операции
        ///             throw new NotImplementedException();
        ///         }
        ///     }
        /// }
    ///
    /// private struct Transition
    /// {
    ///     public TransitionHeader Header;
    ///     public Link Source;
    ///     public Link Linker;
    ///     public Link Target;
    /// }
    ///
    /// </remarks>
    public struct Transition
    {
        public static readonly long Size = StructureHelpers.SizeOf<Transition>();

        public readonly ulong TransactionId;
        public readonly UInt64Link Before;
        public readonly UInt64Link After;
        public readonly Timestamp Timestamp;

        public Transition(UniqueTimestampFactory uniqueTimestampFactory, ulong transactionId, UInt64Link before, UInt64Link after)
        {
            TransactionId = transactionId;
            Before = before;
            After = after;
            Timestamp = uniqueTimestampFactory.Create();
        }

        public Transition(UniqueTimestampFactory uniqueTimestampFactory, ulong transactionId, UInt64Link before)
            : this(uniqueTimestampFactory, transactionId, before, default)
        {
        }

        public Transition(UniqueTimestampFactory uniqueTimestampFactory, ulong transactionId)
            : this(uniqueTimestampFactory, transactionId, default, default)
        {
        }

        public override string ToString() => $"{Timestamp} {TransactionId}: {Before} => {After}";
    }

    /// <remarks>
    /// Другие варианты реализации транзакций (атомарности):
    ///     1. Разделение хранения значения связи ((Source Target) или (Source Linker Target)) и индексов.
    ///     2. Хранение трансформаций/операций в отдельном хранилище Links, но дополнительно потребуется решить вопрос
    ///        со ссылками на внешние идентификаторы, или как-то иначе решить вопрос с пересечениями идентификаторов.
    ///
    /// Где хранить промежуточный список транзакций?
    ///
    /// В оперативной памяти:
    ///  Минусы:
    ///     1. Может усложнить систему, если она будет функционировать самостоятельно,
    ///     так как нужно отдельно выделять память под список трансформаций.
    ///     2. Выделенной оперативной памяти может не хватить, в том случае,
    ///     если транзакция использует слишком много трансформаций.
    ///         -> Можно использовать жёсткий диск для слишком длинных транзакций.
    ///         -> Максимальный размер списка трансформаций можно ограничить / задать константой.
    ///     3. При подтверждении транзакции (Commit) все трансформации записываются разом создавая задержку.
    ///
    /// На жёстком диске:
    ///  Минусы:
    ///     1. Длительный отклик, на запись каждой трансформации.
    ///     2. Лог транзакций дополнительно наполняется отменёнными транзакциями.
    ///         -> Это может решаться упаковкой/исключением дублирующих операций.
    ///         -> Также это может решаться тем, что короткие транзакции вообще
    ///            не будут записываться в случае отката.
    ///     3. Перед тем как выполнять отмену операций транзакции нужно дождаться пока все операции (трансформации)
    ///        будут записаны в лог.
    ///
    /// </remarks>
    public class Transaction : DisposableBase
    {
        private readonly Queue<Transition> _transitions;
        private readonly UInt64LinksTransactionsLayer _layer;
        public bool IsCommitted { get; private set; }
        public bool IsReverted { get; private set; }

        public Transaction(UInt64LinksTransactionsLayer layer)
        {
            _layer = layer;
            if (_layer._currentTransactionId != 0)
            {
                throw new NotSupportedException("Nested transactions not supported.");
            }
            IsCommitted = false;
            IsReverted = false;
            _transitions = new Queue<Transition>();
            SetCurrentTransaction(layer, this);
        }

        public void Commit()
        {
            EnsureTransactionAllowsWriteOperations(this);
            while (_transitions.Count > 0)
            {
                var transition = _transitions.Dequeue();
                _layer._transitions.Enqueue(transition);
            }
            _layer._lastCommitedTransactionId = _layer._currentTransactionId;
            IsCommitted = true;
        }

        private void Revert()
        {
            EnsureTransactionAllowsWriteOperations(this);
            var transitionsToRevert = new Transition[_transitions.Count];
            _transitions.CopyTo(transitionsToRevert, 0);
            for (var i = transitionsToRevert.Length - 1; i >= 0; i--)
            {
                _layer.RevertTransition(transitionsToRevert[i]);
            }
            IsReverted = true;
        }

```

```
            public static void SetCurrentTransaction(UInt64LinksTransactionsLayer layer, Transaction transaction)
            {
                layer._currentTransactionId = layer._lastCommitedTransactionId + 1;
                layer._currentTransactionTransitions = transaction._transitions;
                layer._currentTransaction = transaction;
            }

            public static void EnsureTransactionAllowsWriteOperations(Transaction transaction)
            {
                if (transaction.IsReverted)
                {
                    throw new InvalidOperationException("Transation is reverted.");
                }
                if (transaction.IsCommitted)
                {
                    throw new InvalidOperationException("Transation is commited.");
                }
            }

            protected override void DisposeCore(bool manual, bool wasDisposed)
            {
                if (!wasDisposed && _layer != null && !_layer.IsDisposed)
                {
                    if (!IsCommitted && !IsReverted)
                    {
                        Revert();
                    }
                    _layer.ResetCurrentTransation();
                }
            }

            // TODO: THIS IS EXCEPTION WORKAROUND, REMOVE IT THEN https://github.com/linksplatform/Disposables/issues/13 FIXED
            protected override bool AllowMultipleDisposeCalls => true;
        }

        public static readonly TimeSpan DefaultPushDelay = TimeSpan.FromSeconds(0.1);

        private readonly string _logAddress;
        private readonly FileStream _log;
        private readonly Queue<Transition> _transitions;
        private readonly UniqueTimestampFactory _uniqueTimestampFactory;
        private Task _transitionsPusher;
        private Transition _lastCommitedTransition;
        private ulong _currentTransactionId;
        private Queue<Transition> _currentTransactionTransitions;
        private Transaction _currentTransaction;
        private ulong _lastCommitedTransactionId;

        public UInt64LinksTransactionsLayer(ILinks<ulong> links, string logAddress)
            : base(links)
        {
            if (string.IsNullOrWhiteSpace(logAddress))
            {
                throw new ArgumentNullException(nameof(logAddress));
            }
            // В первой строке файла хранится последняя закоммиченную транзакцию.
            // При запуске это используется для проверки удачного закрытия файла лога.
            // In the first line of the file the last committed transaction is stored.
            // On startup, this is used to check that the log file is successfully closed.
            var lastCommitedTransition = FileHelpers.ReadFirstOrDefault<Transition>(logAddress);
            var lastWrittenTransition = FileHelpers.ReadLastOrDefault<Transition>(logAddress);
            if (!lastCommitedTransition.Equals(lastWrittenTransition))
            {
                Dispose();
                throw new NotSupportedException("Database is damaged, autorecovery is not supported yet.");
            }
            if (lastCommitedTransition.Equals(default(Transition)))
            {
                FileHelpers.WriteFirst(logAddress, lastCommitedTransition);
            }
            _lastCommitedTransition = lastCommitedTransition;
            // TODO: Think about a better way to calculate or store this value
            var allTransitions = FileHelpers.ReadAll<Transition>(logAddress);
            _lastCommitedTransactionId = allTransitions.Max(x => x.TransactionId);
            _uniqueTimestampFactory = new UniqueTimestampFactory();
            _logAddress = logAddress;
            _log = FileHelpers.Append(logAddress);
            _transitions = new Queue<Transition>();
            _transitionsPusher = new Task(TransitionsPusher);
            _transitionsPusher.Start();
        }

        public IList<ulong> GetLinkValue(ulong link) => Links.GetLink(link);

        public override ulong Create()
        {
            var createdLinkIndex = Links.Create();
            var createdLink = new UInt64Link(Links.GetLink(createdLinkIndex));
            CommitTransition(new Transition(_uniqueTimestampFactory, _currentTransactionId, default, createdLink));
            return createdLinkIndex;
        }

        public override ulong Update(IList<ulong> parts)
        {
            var beforeLink = new UInt64Link(Links.GetLink(parts[Constants.IndexPart]));
            parts[Constants.IndexPart] = Links.Update(parts);
            var afterLink = new UInt64Link(Links.GetLink(parts[Constants.IndexPart]));
            CommitTransition(new Transition(_uniqueTimestampFactory, _currentTransactionId, beforeLink, afterLink));
            return parts[Constants.IndexPart];
        }

        public override void Delete(ulong link)
        {
            var deletedLink = new UInt64Link(Links.GetLink(link));
            Links.Delete(link);
            CommitTransition(new Transition(_uniqueTimestampFactory, _currentTransactionId, deletedLink, default));
        }

        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        private Queue<Transition> GetCurrentTransitions() => _currentTransactionTransitions ?? _transitions;

        private void CommitTransition(Transition transition)
        {
            if (_currentTransaction != null)
            {
                Transaction.EnsureTransactionAllowsWriteOperations(_currentTransaction);
            }
            var transitions = GetCurrentTransitions();
            transitions.Enqueue(transition);
        }

        private void RevertTransition(Transition transition)
        {
            if (transition.After.IsNull()) // Revert Deletion with Creation
            {
                Links.Create();
            }
            else if (transition.Before.IsNull()) // Revert Creation with Deletion
            {
                Links.Delete(transition.After.Index);
            }
            else // Revert Update
            {
                Links.Update(new[] { transition.After.Index, transition.Before.Source,
  ↪   transition.Before.Target });
            }
        }

        private void ResetCurrentTransation()
        {
            _currentTransactionId = 0;
            _currentTransactionTransitions = null;
            _currentTransaction = null;
        }

        private void PushTransitions()
        {
            if (_log == null || _transitions == null)
            {
                return;
            }
            for (var i = 0; i < _transitions.Count; i++)
            {
                var transition = _transitions.Dequeue();

                _log.Write(transition);
                _lastCommitedTransition = transition;
            }
        }

        private void TransitionsPusher()
        {
            while (!IsDisposed && _transitionsPusher != null)
            {
                Thread.Sleep(DefaultPushDelay);
                PushTransitions();
            }
        }

        public Transaction BeginTransaction() => new Transaction(this);

        private void DisposeTransitions()
        {
            try
            {
                var pusher = _transitionsPusher;
                if (pusher != null)
                {
                    _transitionsPusher = null;
                    pusher.Wait();
                }
                if (_transitions != null)
                {
                    PushTransitions();
                }
                Disposable.TryDispose(_log);
                FileHelpers.WriteFirst(_logAddress, _lastCommitedTransition);
            }
            catch
            {
            }
        }

        #region DisposalBase

        protected override void DisposeCore(bool manual, bool wasDisposed)
        {
            if (!wasDisposed)
            {
                DisposeTransitions();
            }
            base.DisposeCore(manual, wasDisposed);
        }

        #endregion
    }
}
```

fmt.sh

```
#!/bin/bash
set -e # Exit with nonzero exit code if anything fails

echo """
\\documentclass[7pt,a4paper,fleqn]{report}
\\usepackage[left=6mm,top=5mm,right=5mm,bottom=7mm,landscape]{geometry}
\\textwidth=283mm
\\pagestyle{plain}
\\usepackage[utf8]{inputenc}
\\usepackage[T1]{fontenc}
\\usepackage[T2A]{fontenc}
\\usepackage[gray]{xcolor}
\\usepackage{minted}
```

```
\\makeatletter
\\let\\xUTFviii@two@octets\\UTFviii@two@octets

\\def\\UTFviii@two@octets#1#2{%
\\ifx\\FancyVerbBreakAnywhereBreak#2%
\\expandafter\\xUTFviii@two@octets\\expandafter#1%
\\else
\\xUTFviii@two@octets#1#2%
\\fi
}
\\makeatother

\\usepackage{multicol}
\\usepackage{makeidx}
\\usepackage[columns=2]{idxlayout}
\\makeindex
\\renewcommand{\\thesection}{\\arabic{chapter}.\\arabic{section}}
\\setcounter{chapter}{1}
\\setcounter{section}{0}
\\usepackage[tiny]{titlesec}
\\titlespacing\\chapter{0mm}{0mm}{0mm}
\\titlespacing\\section{0mm}{0mm}{0mm}
\\DeclareUnicodeCharacter{221E}{\\ensuremath{\\infty}}
\\DeclareUnicodeCharacter{FFFD}{\\ensuremath{ }}
\\usepackage{fancyhdr}
\\pagestyle{fancy}
\\fancyhf{}
\\fancyfoot[C]{\\thepage}
\\renewcommand{\\headrulewidth}{0mm}
\\renewcommand{\\footrulewidth}{0mm}
\\renewcommand{\\baselinestretch}{0.7}
\\begin{document}

\\newminted{csharp}{
    breaklines,
    breakanywhere
}

\\sf
\\noindent{\\Large LinksPlatform's Platform.Data.Doublets Class Library}
\\begin{multicols}{2}
"""

# CSharp
#find * -type f -iname '*.cs' -exec sh -c 'enconv "{}"' \;
find . -type f -iname '*.cs' | sort -b
find . -type f -iname '*.cs' | sort -b | python fmt.py

echo """
\\end{multicols}
\\begin{section}{fmt.sh}
\\vspace{2mm}
\\inputminted[tabsize=2,breaklines,linenos=true]{bash}{fmt.sh}
\\end{section}
\\begin{section}{fmt.py}
\\vspace{2mm}
\\inputminted[tabsize=2,breaklines,linenos=true]{python}{fmt.py}
\\end{section}
\\printindex
```

```
\\end{document}
"""
```

fmt.py

```python
#!/usr/bin/python
# -*- coding: utf-8 -*-
import sys
reload(sys)
sys.setdefaultencoding('utf-8')
for line in sys.stdin.readlines():
    line = line.strip()
print "\\index{%s}" % (line.replace('_','\\_'))
print "\\begin{section}{%s}" % (line.replace('_','\\_'))
#print "\\inputminted[tabsize=2,breaklines,linenos=true]{csharp}{%s}" % (line)
print "\\begin{minted}[tabsize=2,breaklines,breakanywhere,linenos=true,xleftmargin=7mm,framesep=4mm,fontsize=\\small,fontfamily=NimbusMono]{csharp}"
f = open(line,"rt")
c = "\n".join([x.strip("\n") for x in f.readlines()])
f.close()
c = c.replace(u'\ufeff','')
print c
print "\\end{minted}"
print "\\end{section}"
print
```

Index

./UInt64LinksTransactionsLayer.cs, 1