h5m.net

page under construction

Logging

There are 2 options for receiving the log information: subscribing to an event or using the LazyService from mst.NET. The later is the preferred one as it provides a weak connection to the library and minimizes the risk of objects remaining in memory by missing the unsubscribe.

The intention is to discontinue the event option in the next versions if there is no request for it.

Log messages via event

The event to be subscribed for receiving the log messages is:

Event for logging

```
                            // Use event for receiving the log messages
                            Messaging.MessagingCenter.LogItemCreated += MessagingCenter_LogItemCreated;

                            [...]

                            Messaging.MessagingCenter.LogItemCreated -= MessagingCenter_LogItemCreated;
```

Log messages via LazyService

Using this option requires the nuget package of mst.NET v1.1.0

Below there is an example on how to use this functionality

Log via LazyService

```
            private static void Main(string[] args)
                {
                                 // initialization of the application
                                _working = true;

                                // or use the lazy service provided by mst.NET
                                Messaging.MessagingCenter.SetLoggingType(true);
                                ServiceLocator.SetLocatorProvider(GetServiceLocator);
                                MessageConsumerTask = Task.Factory.StartNew(ProcessNewMessages);

                                [...]

                                // end of application
                                _working = false;
                                MessageConsumerTask.Wait();
                 }

        #region use LazyService from mst.NET
                 static private bool _working;
                static private Task MessageConsumerTask;
                static LazyService<ILogger> s_loggerService = new LazyService<ILogger>();
                private static ServiceLocatorImpl s_serviceLocator;

                  static private void ProcessNewMessages()
                {
                        LogItem newLogItem;

                        while (_working || !s_loggerService.Value.GeneralMessages.IsEmpty)
                        {
                                try
```

```
                                {
                                        if (s_loggerService.Value.GeneralMessages.IsEmpty)
                                                Thread.Sleep(1);
                                        else if (s_loggerService.Value.GeneralMessages.TryDequeue(out
newLogItem))
                                        {
                                                try
                                                {
                                                        switch (newLogItem.LogLevel)
                                                        {
                                                                case LoggingLevel.Info:
                                                                        Console.WriteLine(newLogItem.
LogLevel + "\t" + newLogItem.Message);
                                                                        break;
                                                                case LoggingLevel.ExceptionsErrors:
                                                                        Console.WriteLine(newLogItem.
LogLevel + "\t" + newLogItem.Message);
                                                                        break;
                                                                case LoggingLevel.Debug:
                                                                        Console.WriteLine(newLogItem.
LogLevel + "\t" + newLogItem.Message);
                                                                        break;
                                                        }
                                                }
                                                catch (Exception except)
                                                {
                                                        Console.WriteLine(except.Message + Environment.
NewLine + except.StackTrace);
                                                }
                                        }
                                }
                                catch (Exception except)
                                {
                                        Console.WriteLine(except.Message + Environment.NewLine + except.
StackTrace);
                                }
                        }
                }


                /// <summary>
                /// Initializes all Application services
                /// </summary>
                private static IServiceLocator GetServiceLocator()
                {
                        if (s_serviceLocator == null)
                        {
                                s_serviceLocator = new ServiceLocatorImpl();
                                s_serviceLocator.Add<ILogger>(new H5MNETLogger());
                        }
                        return s_serviceLocator;
                }
                #endregion
```

Close file

Missing to close the file is a big sources of errors later on in the code. Therefore it is an important operation once all the actions on the file have been completed. It applies for both reading and writing.

The closing of the h5m file is done via the dispose methods. To ensure that the file is closed at a specific time, the Dispose method should be called explicitly.

Reading file

H5M file reader object

All the functionality for reading a h5m file is in the Hdf5FileReader class. Below is an example on creating an object of that type

```
Hdf5FileReader h5MFileReader = Hdf5FileReader.Open(fileName);
```

List of properties

```
// Definition
public void GetAllAttributeNames(string objectName, out List<string> attributeNames)

....
// Use example
List<string> attributeNames;
h5MFileReader.GetAllAttributeNames("/Complete metadata/Base", out attributeNames);
```

Dataset values

Below there is an example on using the method for reading the values of a dataset

```
// Definition
        /// <summary>
        /// Read an 1-dimension dataset
        /// </summary>
        /// <typeparam name="T">the type of the elements in the dataset</typeparam>
        /// <param name="datasetPath">the path to the dataset in the hdf file</param>
        /// <param name="dataSet">an Array containing the values of the dataset</param>
        /// <exception cref="ArgumentException">the datasetPath needs to have a value where to read from<
/exception>
        /// <exception cref="InvalidOperationException">Thrown when it failed ot open the dataset</exception>
        /// <exception cref="IndexOutOfRangeException">the size (length) of the dataset could not be read
properly</exception>
        public void ReadDataset<T>(string datasetPath, out Array dataSet)

// Use example
            Array dataset;
            h5MFileReader.ReadDataset<double>("/Complete metadata/Base", out dataset);
```