LinksPlatform's Platform.Threading Class Library

./ConcurrentQueueExtensions.cs

```csharp
1   using System;
2   using System.Collections.Concurrent;
3   using System.Threading.Tasks;
4   using Platform.Collections.Concurrent;
5
6   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
7
8   namespace Platform.Threading
9   {
10      public static class ConcurrentQueueExtensions
11      {
12          public static async Task AwaitAll(this ConcurrentQueue<Task> queue)
13          {
14              foreach (var item in queue.DequeueAll())
15              {
16                  await item;
17              }
18          }
19
20          public static async Task AwaitOne(this ConcurrentQueue<Task> queue)
21          {
22              if (queue.TryDequeue(out Task item))
23              {
24                  await item;
25              }
26          }
27
28          public static void EnqueueAsRunnedTask(this ConcurrentQueue<Task> queue, Action action)
            => queue.Enqueue(Task.Run(action));
29      }
30  }
```

./Synchronization/ISynchronization.cs

```csharp
1   using System;
2
3   namespace Platform.Threading.Synchronization
4   {
5       /// <summary>
6       /// <para>Represents a synchronization object that supports read and write operations.</para>
7       /// <para>Представляет объект синхронизации с поддержкой операций чтения и записи.</para>
8       /// </summary>
9       public interface ISynchronization
10      {
11          /// <summary>
12          /// <para>Executes action in read access mode.</para>
13          /// <para>Выполняет действие в режиме доступа для чтения.</para>
14          /// </summary>
15          /// <param name="action"><para>The action.</para><para>Действие.</para></param>
16          void ExecuteReadOperation(Action action);
17
18          /// <summary>
19          /// <para>Executes a function in read access mode and returns the function's
            result.</para>
20          /// <para>Выполняет функцию в режиме доступа для чтения и возвращает полученный из неё
            результат.</para>
21          /// </summary>
22          /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
            результата функции.</para></typeparam>
23          /// <param name="function"><para>The function.</para><para>Функция.</para></param>
24          /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
25          TResult ExecuteReadOperation<TResult>(Func<TResult> function);
26
27          /// <summary>
28          /// <para>Executes action in write access mode.</para>
29          /// <para>Выполняет действие в режиме доступа для записи.</para>
30          /// </summary>
31          /// <param name="action"><para>The action.</para><para>Действие.</para></param>
32          void ExecuteWriteOperation(Action action);
33
34          /// <summary>
35          /// <para>Executes a function in write access mode and returns the function's
            result.</para>
36          /// <para>Выполняет функцию в режиме доступа для записи и возвращает полученный из неё
            результат.</para>
37          /// </summary>
38          /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
            результата функции.</para></typeparam>
```

```csharp
        /// <param name="function"><para>The function.</para><para>Функция.</para></param>
        /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
        TResult ExecuteWriteOperation<TResult>(Func<TResult> function);
    }
}
```

./Synchronization/ISynchronizationExtensions.cs

```csharp
using System;

namespace Platform.Threading.Synchronization
{
    /// <summary>
    /// <para>Contains extension methods for the <see cref="ISynchronization"/> interface.</para>
    /// <para>Содержит методы расширения для интерфейса <see cref="ISynchronization"/>.</para>
    /// </summary>
    public static class ISynchronizationExtensions
    {
        /// <include file='bin\Release\netstandard2.0\Documentation.xml'
        ///    path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
        ///    n.ExecuteReadOperation``1(System.Func{``0})"]/*'/>
        /// <typeparam name="TParam"><para>The parameter type.</para><para>Тип
        ///    параметра.</para></typeparam>
        /// <param name="synchronization"><para>Synchronization
        ///    object.</para><para>Синхронизация объекта.</para></param>
        /// <param name="parameter"><para>The parameter</para><para>Параметр.</para></param>
        public static TResult ExecuteReadOperation<TResult, TParam>(this ISynchronization
            synchronization, TParam parameter, Func<TParam, TResult> function) =>
            synchronization.ExecuteReadOperation(() => function(parameter));

        /// <include file='bin\Release\netstandard2.0\Documentation.xml'
        ///    path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
        ///    n.ExecuteReadOperation(System.Action)"]/*'/>
        /// <typeparam name="TParam"><para>The parameter type.</para><para>Тип
        ///    параметра.</para></typeparam>
        /// <param name="synchronization"><para>Synchronization
        ///    object.</para><para>Синхронизация объекта.</para></param>
        /// <param name="parameter"><para>The parameter</para><para>Параметр.</para></param>
        public static void ExecuteReadOperation<TParam>(this ISynchronization synchronization,
            TParam parameter, Action<TParam> action) => synchronization.ExecuteReadOperation(()
            => action(parameter));

        /// <include file='bin\Release\netstandard2.0\Documentation.xml'
        ///    path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
        ///    n.ExecuteWriteOperation``1(System.Func{``0})"]/*'/>
        /// <typeparam name="TParam"><para>The parameter type.</para><para>Тип
        ///    параметра.</para></typeparam>
        /// <param name="synchronization"><para>Synchronization
        ///    object.</para><para>Синхронизация объекта.</para></param>
        /// <param name="parameter"><para>The parameter</para><para>Параметр.</para></param>
        public static TResult ExecuteWriteOperation<TResult, TParam>(this ISynchronization
            synchronization, TParam parameter, Func<TParam, TResult> function) =>
            synchronization.ExecuteWriteOperation(() => function(parameter));

        /// <include file='bin\Release\netstandard2.0\Documentation.xml'
        ///    path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
        ///    n.ExecuteWriteOperation(System.Action)"]/*'/>
        /// <typeparam name="TParam"><para>The parameter type.</para><para>Тип
        ///    параметра.</para></typeparam>
        /// <param name="synchronization"><para>Synchronization
        ///    object.</para><para>Синхронизация объекта.</para></param>
        /// <param name="parameter"><para>The parameter</para><para>Параметр.</para></param>
        public static void ExecuteWriteOperation<TParam>(this ISynchronization synchronization,
            TParam parameter, Action<TParam> action) => synchronization.ExecuteWriteOperation(()
            => action(parameter));

        /// <include file='bin\Release\netstandard2.0\Documentation.xml'
        ///    path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
        ///    n.ExecuteReadOperation``1(System.Func{``0})"]/*'/>
        /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
        ///    параметра.</para></typeparam>
        /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
        ///    параметра.</para></typeparam>
        /// <param name="synchronization"><para>Synchronization
        ///    object.</para><para>Синхронизация объекта.</para></param>
        /// <param name="parameter1"><para>The first parameter</para><para>Первый
        ///    параметр.</para></param>
```

```csharp
40        /// <param name="parameter2"><para>The second parameter</para><para>Второй
      ↪  параметр.</para></param>
41        public static TResult ExecuteReadOperation<TResult, TParam1, TParam2>(this
      ↪  ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2,
      ↪  Func<TParam1, TParam2, TResult> function) => synchronization.ExecuteReadOperation(()
      ↪  => function(parameter1, parameter2));
42
43        /// <include file='bin\Release\netstandard2.0\Documentation.xml'
      ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio⌋
      ↪  n.ExecuteReadOperation(System.Action)"]/*'/>
44        /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
      ↪  параметра.</para></typeparam>
45        /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
      ↪  параметра.</para></typeparam>
46        /// <param name="synchronization"><para>Synchronization
      ↪  object.</para><para>Синхронизация объекта.</para></param>
47        /// <param name="parameter1"><para>The first parameter</para><para>Первый
      ↪  параметр.</para></param>
48        /// <param name="parameter2"><para>The second parameter</para><para>Второй
      ↪  параметр.</para></param>
49        public static void ExecuteReadOperation<TParam1, TParam2>(this ISynchronization
      ↪  synchronization, TParam1 parameter1, TParam2 parameter2, Action<TParam1, TParam2>
      ↪  action) => synchronization.ExecuteReadOperation(() => action(parameter1,
      ↪  parameter2));
50
51        /// <include file='bin\Release\netstandard2.0\Documentation.xml'
      ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio⌋
      ↪  n.ExecuteWriteOperation``1(System.Func{``0})"]/*'/>
52        /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
      ↪  параметра.</para></typeparam>
53        /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
      ↪  параметра.</para></typeparam>
54        /// <param name="synchronization"><para>Synchronization
      ↪  object.</para><para>Синхронизация объекта.</para></param>
55        /// <param name="parameter1"><para>The first parameter</para><para>Первый
      ↪  параметр.</para></param>
56        /// <param name="parameter2"><para>The second parameter</para><para>Второй
      ↪  параметр.</para></param>
57        public static TResult ExecuteWriteOperation<TResult, TParam1, TParam2>(this
      ↪  ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2,
      ↪  Func<TParam1, TParam2, TResult> function) =>
      ↪  synchronization.ExecuteWriteOperation(() => function(parameter1, parameter2));
58
59        /// <include file='bin\Release\netstandard2.0\Documentation.xml'
      ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio⌋
      ↪  n.ExecuteWriteOperation(System.Action)"]/*'/>
60        /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
      ↪  параметра.</para></typeparam>
61        /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
      ↪  параметра.</para></typeparam>
62        /// <param name="synchronization"><para>Synchronization
      ↪  object.</para><para>Синхронизация объекта.</para></param>
63        /// <param name="parameter1"><para>The first parameter</para><para>Первый
      ↪  параметр.</para></param>
64        /// <param name="parameter2"><para>The second parameter</para><para>Второй
      ↪  параметр.</para></param>
65        public static void ExecuteWriteOperation<TParam1, TParam2>(this ISynchronization
      ↪  synchronization, TParam1 parameter1, TParam2 parameter2, Action<TParam1, TParam2>
      ↪  action) => synchronization.ExecuteWriteOperation(() => action(parameter1,
      ↪  parameter2));
66
67        /// <include file='bin\Release\netstandard2.0\Documentation.xml'
      ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio⌋
      ↪  n.ExecuteReadOperation``1(System.Func{``0})"]/*'/>
68        /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
      ↪  параметра.</para></typeparam>
69        /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
      ↪  параметра.</para></typeparam>
70        /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
      ↪  параметра.</para></typeparam>
71        /// <param name="synchronization"><para>Synchronization
      ↪  object.</para><para>Синхронизация объекта.</para></param>
72        /// <param name="parameter1"><para>The first parameter</para><para>Первый
      ↪  параметр.</para></param>
```

```csharp
        /// <param name="parameter2"><para>The second parameter</para><para>Второй
        ↪ параметр.</para></param>
        /// <param name="parameter3"><para>The third parameter</para><para>Третий
        ↪ параметр.</para></param>
        public static TResult ExecuteReadOperation<TResult, TParam1, TParam2, TParam3>(this
            ↪ ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2, TParam3
            ↪ parameter3, Func<TParam1, TParam2, TParam3, TResult> function) =>
            ↪ synchronization.ExecuteReadOperation(() => function(parameter1, parameter2,
            ↪ parameter3));

        /// <include file='bin\Release\netstandard2.0\Documentation.xml'
        ↪ path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio⌋
        ↪ n.ExecuteReadOperation(System.Action)"]/*'/>
        /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
        ↪ параметра.</para></typeparam>
        /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
        ↪ параметра.</para></typeparam>
        /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
        ↪ параметра.</para></typeparam>
        /// <param name="synchronization"><para>Synchronization
        ↪ object.</para><para>Синхронизация объекта.</para></param>
        /// <param name="parameter1"><para>The first parameter</para><para>Первый
        ↪ параметр.</para></param>
        /// <param name="parameter2"><para>The second parameter</para><para>Второй
        ↪ параметр.</para></param>
        /// <param name="parameter3"><para>The third parameter</para><para>Третий
        ↪ параметр.</para></param>
        public static void ExecuteReadOperation<TParam1, TParam2, TParam3>(this ISynchronization
            ↪ synchronization, TParam1 parameter1, TParam2 parameter2, TParam3 parameter3,
            ↪ Action<TParam1, TParam2, TParam3> action) => synchronization.ExecuteReadOperation(()
            ↪ => action(parameter1, parameter2, parameter3));

        /// <include file='bin\Release\netstandard2.0\Documentation.xml'
        ↪ path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio⌋
        ↪ n.ExecuteWriteOperation``1(System.Func{``0})"]/*'/>
        /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
        ↪ параметра.</para></typeparam>
        /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
        ↪ параметра.</para></typeparam>
        /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
        ↪ параметра.</para></typeparam>
        /// <param name="synchronization"><para>Synchronization
        ↪ object.</para><para>Синхронизация объекта.</para></param>
        /// <param name="parameter1"><para>The first parameter</para><para>Первый
        ↪ параметр.</para></param>
        /// <param name="parameter2"><para>The second parameter</para><para>Второй
        ↪ параметр.</para></param>
        /// <param name="parameter3"><para>The third parameter</para><para>Третий
        ↪ параметр.</para></param>
        public static TResult ExecuteWriteOperation<TResult, TParam1, TParam2, TParam3>(this
            ↪ ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2, TParam3
            ↪ parameter3, Func<TParam1, TParam2, TParam3, TResult> function) =>
            ↪ synchronization.ExecuteWriteOperation(() => function(parameter1, parameter2,
            ↪ parameter3));

        /// <include file='bin\Release\netstandard2.0\Documentation.xml'
        ↪ path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio⌋
        ↪ n.ExecuteWriteOperation(System.Action)"]/*'/>
        /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
        ↪ параметра.</para></typeparam>
        /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
        ↪ параметра.</para></typeparam>
        /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
        ↪ параметра.</para></typeparam>
        /// <param name="synchronization"><para>Synchronization
        ↪ object.</para><para>Синхронизация объекта.</para></param>
        /// <param name="parameter1"><para>The first parameter</para><para>Первый
        ↪ параметр.</para></param>
        /// <param name="parameter2"><para>The second parameter</para><para>Второй
        ↪ параметр.</para></param>
        /// <param name="parameter3"><para>The third parameter</para><para>Третий
        ↪ параметр.</para></param>
```

```csharp
105      public static void ExecuteWriteOperation<TParam1, TParam2, TParam3>(this
    ↪  ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2, TParam3
    ↪  parameter3, Action<TParam1, TParam2, TParam3> action) =>
    ↪  synchronization.ExecuteWriteOperation(() => action(parameter1, parameter2,
    ↪  parameter3));
106
107      /// <include file='bin\Release\netstandard2.0\Documentation.xml'
    ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio⌋
    ↪  n.ExecuteReadOperation``1(System.Func{``0})"]/*'/>
108      /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
    ↪  параметра.</para></typeparam>
109      /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
    ↪  параметра.</para></typeparam>
110      /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
    ↪  параметра.</para></typeparam>
111      /// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
    ↪  параметра.</para></typeparam>
112      /// <param name="synchronization"><para>Synchronization
    ↪  object.</para><para>Синхронизация объекта.</para></param>
113      /// <param name="parameter1"><para>The first parameter</para><para>Первый
    ↪  параметр.</para></param>
114      /// <param name="parameter2"><para>The second parameter</para><para>Второй
    ↪  параметр.</para></param>
115      /// <param name="parameter3"><para>The third parameter</para><para>Третий
    ↪  параметр.</para></param>
116      /// <param name="parameter4"><para>The forth parameter</para><para>Чертвёртый
    ↪  параметр.</para></param>
117      public static TResult ExecuteReadOperation<TResult, TParam1, TParam2, TParam3,
    ↪  TParam4>(this ISynchronization synchronization, TParam1 parameter1, TParam2
    ↪  parameter2, TParam3 parameter3, TParam4 parameter4, Func<TParam1, TParam2, TParam3,
    ↪  TParam4, TResult> function) => synchronization.ExecuteReadOperation(() =>
    ↪  function(parameter1, parameter2, parameter3, parameter4));
118
119      /// <include file='bin\Release\netstandard2.0\Documentation.xml'
    ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio⌋
    ↪  n.ExecuteReadOperation(System.Action)"]/*'/>
120      /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
    ↪  параметра.</para></typeparam>
121      /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
    ↪  параметра.</para></typeparam>
122      /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
    ↪  параметра.</para></typeparam>
123      /// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
    ↪  параметра.</para></typeparam>
124      /// <param name="synchronization"><para>Synchronization
    ↪  object.</para><para>Синхронизация объекта.</para></param>
125      /// <param name="parameter1"><para>The first parameter</para><para>Первый
    ↪  параметр.</para></param>
126      /// <param name="parameter2"><para>The second parameter</para><para>Второй
    ↪  параметр.</para></param>
127      /// <param name="parameter3"><para>The third parameter</para><para>Третий
    ↪  параметр.</para></param>
128      /// <param name="parameter4"><para>The forth parameter</para><para>Чертвёртый
    ↪  параметр.</para></param>
129      public static void ExecuteReadOperation<TParam1, TParam2, TParam3, TParam4>(this
    ↪  ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2, TParam3
    ↪  parameter3, TParam4 parameter4, Action<TParam1, TParam2, TParam3, TParam4> action)
    ↪  => synchronization.ExecuteReadOperation(() => action(parameter1, parameter2,
    ↪  parameter3, parameter4));
130
131      /// <include file='bin\Release\netstandard2.0\Documentation.xml'
    ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio⌋
    ↪  n.ExecuteWriteOperation``1(System.Func{``0})"]/*'/>
132      /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
    ↪  параметра.</para></typeparam>
133      /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
    ↪  параметра.</para></typeparam>
134      /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
    ↪  параметра.</para></typeparam>
135      /// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
    ↪  параметра.</para></typeparam>
136      /// <param name="synchronization"><para>Synchronization
    ↪  object.</para><para>Синхронизация объекта.</para></param>
137      /// <param name="parameter1"><para>The first parameter</para><para>Первый
    ↪  параметр.</para></param>
```

```csharp
138         /// <param name="parameter2"><para>The second parameter</para><para>Второй
     ↪  параметр.</para></param>
139         /// <param name="parameter3"><para>The third parameter</para><para>Третий
     ↪  параметр.</para></param>
140         /// <param name="parameter4"><para>The forth parameter</para><para>Чертвёртый
     ↪  параметр.</para></param>
141         public static TResult ExecuteWriteOperation<TResult, TParam1, TParam2, TParam3,
     ↪  TParam4>(this ISynchronization synchronization, TParam1 parameter1, TParam2
     ↪  parameter2, TParam3 parameter3, TParam4 parameter4, Func<TParam1, TParam2, TParam3,
     ↪  TParam4, TResult> function) => synchronization.ExecuteWriteOperation(() =>
     ↪  function(parameter1, parameter2, parameter3, parameter4));
142
143         /// <include file='bin\Release\netstandard2.0\Documentation.xml'
     ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio⌋
     ↪  n.ExecuteWriteOperation(System.Action)"]/*'/>
144         /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
     ↪  параметра.</para></typeparam>
145         /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
     ↪  параметра.</para></typeparam>
146         /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
     ↪  параметра.</para></typeparam>
147         /// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
     ↪  параметра.</para></typeparam>
148         /// <param name="synchronization"><para>Synchronization
     ↪  object.</para><para>Синхронизация объекта.</para></param>
149         /// <param name="parameter1"><para>The first parameter</para><para>Первый
     ↪  параметр.</para></param>
150         /// <param name="parameter2"><para>The second parameter</para><para>Второй
     ↪  параметр.</para></param>
151         /// <param name="parameter3"><para>The third parameter</para><para>Третий
     ↪  параметр.</para></param>
152         /// <param name="parameter4"><para>The forth parameter</para><para>Чертвёртый
     ↪  параметр.</para></param>
153         public static void ExecuteWriteOperation<TParam1, TParam2, TParam3, TParam4>(this
     ↪  ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2, TParam3
     ↪  parameter3, TParam4 parameter4, Action<TParam1, TParam2, TParam3, TParam4> action)
     ↪  => synchronization.ExecuteWriteOperation(() => action(parameter1, parameter2,
     ↪  parameter3, parameter4));
154     }
155 }
```

./Synchronization/ISynchronized.cs

```csharp
1  namespace Platform.Threading.Synchronization
2  {
3      /// <summary>
4      /// <para>Represents extendable synchronized interface access gate.</para>
5      /// <para>Представляет расширяемый интерфейс шлюза синхронизированного доступа.</para>
6      /// </summary>
7      /// <typeparam name="TInterface"><para>Synchronized interface.</para><para>Синхронизируемый
     ↪  интерфейс.</para></typeparam>
8      public interface ISynchronized<out TInterface>
9      {
10         /// <summary>
11         /// <para>Gets sychronization method.</para>
12         /// <para>Возвращает способ синхронизации.</para>
13         /// </summary>
14         ISynchronization SyncRoot { get; }
15
16         /// <summary>
17         /// <para>Get source version of <typeparamref name="TInterface"/>, that does not
     ↪  garantee thread safe access synchronization.</para>
18         /// <para>Возвращает исходную версию <typeparamref name="TInterface"/>, которая не
     ↪  гарантирует потокобезопасную синхронизацию доступа.</para>
19         /// </summary>
20         /// <remarks>
21         /// <para>It is unsafe to use it directly, unless compound context using SyncRoot is
     ↪  created.</para>
22         /// <para>Использовать напрямую небезопасно, за исключением ситуации когда создаётся
     ↪  составной контекст с использованием SyncRoot.</para>
23         /// </remarks>
24         TInterface Unsync { get; }
25
26         /// <summary>
27         /// <para>Get wrapped/decorated version of <typeparamref name="TInterface"/>, that does
     ↪  garantee thread safe access synchronization.</para>
```

```csharp
            /// <para>Возвращает обернутую/декорированную версию <typeparamref name="TInterface"/>,
            ↪  которая гарантирует потокобезопасную синхронизацию доступа.</para>
            /// </summary>
            /// <remarks>
            /// <para>It is safe to use it directly, because it must be thread safe
            ↪  implementation.</para>
            /// <para>Безопасно использовать напрямую, так как реализация должна быть
            ↪  потокобезопасной.</para>
            /// </remarks>
            TInterface Sync { get; }
        }
    }
```

./Synchronization/ReaderWriterLockSynchronization.cs

```csharp
using System;
using System.Threading;

namespace Platform.Threading.Synchronization
{
    /// <summary>
    /// Implementation of <see cref="ISynchronization"/> based on <see
    ↪  cref="ReaderWriterLockSlim"/>.
    /// </summary>
    public class ReaderWriterLockSynchronization : ISynchronization
    {
        private readonly ReaderWriterLockSlim _rwLock = new
        ↪  ReaderWriterLockSlim(LockRecursionPolicy.SupportsRecursion);

        /// <include file='bin\Release\netstandard2.0\Documentation.xml'
        ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio⌐
        ↪  n.ExecuteReadOperation(System.Action)"]/*'/>
        public void ExecuteReadOperation(Action action)
        {
            _rwLock.EnterReadLock();
            try
            {
                action();
            }
            finally
            {
                _rwLock.ExitReadLock();
            }
        }

        /// <include file='bin\Release\netstandard2.0\Documentation.xml'
        ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio⌐
        ↪  n.ExecuteReadOperation``1(System.Func{``0})"]/*'/>
        public TResult ExecuteReadOperation<TResult>(Func<TResult> function)
        {
            _rwLock.EnterReadLock();
            try
            {
                return function();
            }
            finally
            {
                _rwLock.ExitReadLock();
            }
        }

        /// <include file='bin\Release\netstandard2.0\Documentation.xml'
        ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio⌐
        ↪  n.ExecuteWriteOperation(System.Action)"]/*'/>
        public void ExecuteWriteOperation(Action action)
        {
            _rwLock.EnterWriteLock();
            try
            {
                action();
            }
            finally
            {
                _rwLock.ExitWriteLock();
            }
        }
```

```
55          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
        ↪   path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio⌋
        ↪   n.ExecuteWriteOperation``1(System.Func{``0})"]/*'/>
56          public TResult ExecuteWriteOperation<TResult>(Func<TResult> function)
57          {
58              _rwLock.EnterWriteLock();
59              try
60              {
61                  return function();
62              }
63              finally
64              {
65                  _rwLock.ExitWriteLock();
66              }
67          }
68      }
69  }
```

./Synchronization/Unsynchronization.cs

```
1   using System;
2
3   namespace Platform.Threading.Synchronization
4   {
5       /// <summary>
6       /// Implementation of <see cref="ISynchronization"/> that makes no actual synchronization.
7       /// </summary>
8       public class Unsynchronization : ISynchronization
9       {
10          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
        ↪   path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio⌋
        ↪   n.ExecuteReadOperation(System.Action)"]/*'/>
11          public void ExecuteReadOperation(Action action) => action();
12
13          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
        ↪   path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio⌋
        ↪   n.ExecuteReadOperation``1(System.Func{``0})"]/*'/>
14          public TResult ExecuteReadOperation<TResult>(Func<TResult> function) => function();
15
16          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
        ↪   path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio⌋
        ↪   n.ExecuteWriteOperation(System.Action)"]/*'/>
17          public void ExecuteWriteOperation(Action action) => action();
18
19          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
        ↪   path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio⌋
        ↪   n.ExecuteWriteOperation``1(System.Func{``0})"]/*'/>
20          public TResult ExecuteWriteOperation<TResult>(Func<TResult> function) => function();
21      }
22  }
```

./TaskExtensions.cs

```
1   using System.Threading.Tasks;
2
3   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
4
5   namespace Platform.Threading
6   {
7       public static class TaskExtensions
8       {
9           public static T AwaitResult<T>(this Task<T> task) => task.GetAwaiter().GetResult();
10      }
11  }
```

./ThreadHelpers.cs

```
1   using System;
2   using System.Threading;
3
4   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
5
6   namespace Platform.Threading
7   {
8       public static class ThreadHelpers
9       {
10          public static readonly int DefaultMaxStackSize;
11          public static readonly int ExtendedMaxStackSize = 200 * 1024 * 1024;
12          public static readonly int DefaultSleepTimeout = 1;
13
14          public static void SyncInvokeWithExtendedStack<T>(T param, Action<object> action) =>
        ↪   SyncInvokeWithExtendedStack(param, action, ExtendedMaxStackSize);
```

```csharp
        public static void SyncInvokeWithExtendedStack<T>(T param, Action<object> action, int
        ↪  maxStackSize) => StartNew(param, action, maxStackSize).Join();

        public static void SyncInvokeWithExtendedStack(Action action) =>
        ↪  SyncInvokeWithExtendedStack(action, ExtendedMaxStackSize);

        public static void SyncInvokeWithExtendedStack(Action action, int maxStackSize) =>
        ↪  StartNew(action, maxStackSize).Join();

        public static Thread StartNew<T>(T param, Action<object> action) => StartNew(param,
        ↪  action, DefaultMaxStackSize);

        public static Thread StartNew<T>(T param, Action<object> action, int maxStackSize)
        {
            var thread = new Thread(new ParameterizedThreadStart(action), maxStackSize);
            thread.Start(param);
            return thread;
        }

        public static Thread StartNew(Action action) => StartNew(action, DefaultMaxStackSize);

        public static Thread StartNew(Action action, int maxStackSize)
        {
            var thread = new Thread(new ThreadStart(action), maxStackSize);
            thread.Start();
            return thread;
        }

        public static void Sleep() => Thread.Sleep(DefaultSleepTimeout);
    }
}
```

Index