

LinksPlatform's Platform.Threading Class Library

./ConcurrentQueueExtensions.cs

```
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)
29             => queue.Enqueue(Task.Run(action));
30     }
31 }
```

./Synchronization/ISynchronization.cs

```
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
20         ///     => result.</para>
21         /// <para>Выполняет функцию в режиме доступа для чтения и возвращает полученный из неё
22         ///     => результат.</para>
23         /// </summary>
24         /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
25         ///     => результата функции.</para></typeparam>
26         /// <param name="function"><para>The function.</para><para>Функция.</para></param>
27         /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
28         TResult ExecuteReadOperation<TResult>(Func<TResult> function);
29
30         /// <summary>
31         /// <para>Executes action in write access mode.</para>
32         /// <para>Выполняет действие в режиме доступа для записи.</para>
33         /// </summary>
34         /// <param name="action"><para>The action.</para><para>Действие.</para></param>
35         void ExecuteWriteOperation(Action action);
36
37         /// <summary>
38         /// <para>Executes a function in write access mode and returns the function's
39         ///     => result.</para>
40         /// <para>Выполняет функцию в режиме доступа для записи и возвращает полученный из неё
41         ///     => результат.</para>
42         /// </summary>
43         /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
44         ///     => результата функции.</para></typeparam>
45     }
46 }
```

```

39     /// <param name="function"><para>The function.</para><para>Функция.</para></param>
40     /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
41     TResult ExecuteWriteOperation<TResult>(Func<TResult> function);
42 }
43 }

```

./Synchronization/ISynchronizationExtensions.cs

```

1  using System;
2
3  namespace Platform.Threading.Synchronization
4  {
5      /// <summary>
6      /// <para>Contains extension methods for the <see cref="ISynchronization"/> interface.</para>
7      /// <para>Содержит методы расширения для интерфейса <see cref="ISynchronization"/>.</para>
8      /// </summary>
9      public static class ISynchronizationExtensions
10     {
11         /// <summary>
12         /// <para>Executes a function in read access mode and returns the function's
13         → result.</para>
14         /// <para>Выполняет функцию в режиме доступа для чтения и возвращает полученный из неё
15         → результат.</para>
16         /// </summary>
17         /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
18         → результата функции.</para></typeparam>
19         /// <typeparam name="TParam"><para>The parameter type.</para><para>Тип
20         → параметра.</para></typeparam>
21         /// <param name="synchronization"><para>Synchronization
22         → object.</para><para>Синхронизация объекта.</para></param>
23         /// <param name="parameter"><para>The parameter</para><para>Параметр.</para></param>
24         /// <param name="function"><para>The function.</para><para>Функция.</para></param>
25         /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
26         public static TResult ExecuteReadOperation<TResult, TParam>(this ISynchronization
27         → synchronization, TParam parameter, Func<TParam, TResult> function) =>
28         → synchronization.ExecuteReadOperation(() => function(parameter));
29
30         /// <summary>
31         /// <para>Executes action in read access mode.</para>
32         /// <para>Выполняет действие в режиме доступа для чтения.</para>
33         /// </summary>
34         /// <typeparam name="TParam"><para>The parameter type.</para><para>Тип
35         → параметра.</para></typeparam>
36         /// <param name="synchronization"><para>Synchronization
37         → object.</para><para>Синхронизация объекта.</para></param>
38         /// <param name="parameter"><para>The parameter</para><para>Параметр.</para></param>
39         /// <param name="action"><para>The action.</para><para>Действие.</para></param>
40         public static void ExecuteReadOperation<TParam>(this ISynchronization synchronization,
41         → TParam parameter, Action<TParam> action) => synchronization.ExecuteReadOperation(()
42         → => action(parameter));
43
44         /// <summary>
45         /// <para>Executes a function in write access mode and returns the function's
46         → result.</para>
47         /// <para>Выполняет функцию в режиме доступа для записи и возвращает полученный из неё
48         → результат.</para>
49         /// </summary>
50         /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
51         → результата функции.</para></typeparam>
52         /// <typeparam name="TParam"><para>The parameter type.</para><para>Тип
53         → параметра.</para></typeparam>
54         /// <param name="synchronization"><para>Synchronization
55         → object.</para><para>Синхронизация объекта.</para></param>
56         /// <param name="parameter"><para>The parameter</para><para>Параметр.</para></param>
57         /// <param name="function"><para>The function.</para><para>Функция.</para></param>
58         /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
59         public static TResult ExecuteWriteOperation<TResult, TParam>(this ISynchronization
60         → synchronization, TParam parameter, Func<TParam, TResult> function) =>
61         → synchronization.ExecuteWriteOperation(() => function(parameter));
62
63         /// <summary>
64         /// <para>Executes action in write access mode.</para>
65         /// <para>Выполняет действие в режиме доступа для записи.</para>
66         /// </summary>
67         /// <typeparam name="TParam"><para>The parameter type.</para><para>Тип
68         → параметра.</para></typeparam>
69         /// <param name="synchronization"><para>Synchronization
70         → object.</para><para>Синхронизация объекта.</para></param>
71         /// <param name="parameter"><para>The parameter</para><para>Параметр.</para></param>
72         /// <param name="action"><para>The action.</para><para>Действие.</para></param>
73         public static void ExecuteWriteOperation<TParam>(this ISynchronization synchronization,
74         → TParam parameter, Action<TParam> action) => synchronization.ExecuteWriteOperation(()
75         → => action(parameter));
76     }
77 }

```

```

51     /// <param name="parameter"><para>The parameter.</para><para>Параметр.</para></param>
52     /// <param name="action"><para>The action.</para><para>Действие.</para></param>
53     public static void ExecuteWriteOperation<TParam>(this ISynchronization synchronization,
    → TParam parameter, Action<TParam> action) => synchronization.ExecuteWriteOperation((
    → => action(parameter));
54
55     /// <summary>
56     /// <para>Executes a function in read access mode and returns the function's
    → result.</para>
57     /// <para>Выполняет функцию в режиме доступа для чтения и возвращает полученный из неё
    → результат.</para>
58     /// </summary>
59     /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
    → результата функции.</para></typeparam>
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     /// <param name="function"><para>The function.</para><para>Функция.</para></param>
66     /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
67     public static TResult ExecuteReadOperation<TResult, TParam1, TParam2>(this
    → ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2,
    → Func<TParam1, TParam2, TResult> function) => synchronization.ExecuteReadOperation((
    → => function(parameter1, parameter2));
68
69     /// <summary>
70     /// <para>Executes action in read access mode.</para>
71     /// <para>Выполняет действие в режиме доступа для чтения.</para>
72     /// </summary>
73     /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
    → параметра.</para></typeparam>
74     /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
    → параметра.</para></typeparam>
75     /// <param name="synchronization"><para>Synchronization
    → object.</para><para>Синхронизация объекта.</para></param>
76     /// <param name="parameter1"><para>The first parameter</para><para>Первый
    → параметр.</para></param>
77     /// <param name="parameter2"><para>The second parameter</para><para>Второй
    → параметр.</para></param>
78     /// <param name="action"><para>The action.</para><para>Действие.</para></param>
79     public static void ExecuteReadOperation<TParam1, TParam2>(this ISynchronization
    → synchronization, TParam1 parameter1, TParam2 parameter2, Action<TParam1, TParam2>
    → action) => synchronization.ExecuteReadOperation(() => action(parameter1,
    → parameter2));
80
81     /// <summary>
82     /// <para>Executes a function in write access mode and returns the function's
    → result.</para>
83     /// <para>Выполняет функцию в режиме доступа для записи и возвращает полученный из неё
    → результат.</para>
84     /// </summary>
85     /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
    → результата функции.</para></typeparam>
86     /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
    → параметра.</para></typeparam>
87     /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
    → параметра.</para></typeparam>
88     /// <param name="synchronization"><para>Synchronization
    → object.</para><para>Синхронизация объекта.</para></param>
89     /// <param name="parameter1"><para>The first parameter</para><para>Первый
    → параметр.</para></param>
90     /// <param name="parameter2"><para>The second parameter</para><para>Второй
    → параметр.</para></param>
91     /// <param name="function"><para>The function.</para><para>Функция.</para></param>
92     /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
93     public static TResult ExecuteWriteOperation<TResult, TParam1, TParam2>(this
    → ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2,
    → Func<TParam1, TParam2, TResult> function) =>
    → synchronization.ExecuteWriteOperation(() => function(parameter1, parameter2));
94

```

```

95  /// <summary>
96  /// <para>Executes action in write access mode.</para>
97  /// <para>Выполняет действие в режиме доступа для записи.</para>
98  /// </summary>
99  /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
   → параметра.</para></typeparam>
100 /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
   → параметра.</para></typeparam>
101 /// <param name="synchronization"><para>Synchronization
   → object.</para><para>Синхронизация объекта.</para></param>
102 /// <param name="parameter1"><para>The first parameter</para><para>Первый
   → параметр.</para></param>
103 /// <param name="parameter2"><para>The second parameter</para><para>Второй
   → параметр.</para></param>
104 /// <param name="action"><para>The action.</para><para>Действие.</para></param>
105 public static void ExecuteWriteOperation<TParam1, TParam2>(this ISynchronization
   → synchronization, TParam1 parameter1, TParam2 parameter2, Action<TParam1, TParam2>
   → action) => synchronization.ExecuteWriteOperation(() => action(parameter1,
   → parameter2));
106
107 /// <summary>
108 /// <para>Executes a function in read access mode and returns the function's
   → result.</para>
109 /// <para>Выполняет функцию в режиме доступа для чтения и возвращает полученный из неё
   → результат.</para>
110 /// </summary>
111 /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
   → результата функции.</para></typeparam>
112 /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
   → параметра.</para></typeparam>
113 /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
   → параметра.</para></typeparam>
114 /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
   → параметра.</para></typeparam>
115 /// <param name="synchronization"><para>Synchronization
   → object.</para><para>Синхронизация объекта.</para></param>
116 /// <param name="parameter1"><para>The first parameter</para><para>Первый
   → параметр.</para></param>
117 /// <param name="parameter2"><para>The second parameter</para><para>Второй
   → параметр.</para></param>
118 /// <param name="parameter3"><para>The third parameter</para><para>Третий
   → параметр.</para></param>
119 /// <param name="function"><para>The function.</para><para>Функция.</para></param>
120 /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
121 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));
122
123 /// <summary>
124 /// <para>Executes action in read access mode.</para>
125 /// <para>Выполняет действие в режиме доступа для чтения.</para>
126 /// </summary>
127 /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
   → параметра.</para></typeparam>
128 /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
   → параметра.</para></typeparam>
129 /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
   → параметра.</para></typeparam>
130 /// <param name="synchronization"><para>Synchronization
   → object.</para><para>Синхронизация объекта.</para></param>
131 /// <param name="parameter1"><para>The first parameter</para><para>Первый
   → параметр.</para></param>
132 /// <param name="parameter2"><para>The second parameter</para><para>Второй
   → параметр.</para></param>
133 /// <param name="parameter3"><para>The third parameter</para><para>Третий
   → параметр.</para></param>
134 /// <param name="action"><para>The action.</para><para>Действие.</para></param>
135 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));
136
137 /// <summary>

```

```

138  /// <para>Executes a function in write access mode and returns the function's
139  → result.</para>
140  /// <para>Выполняет функцию в режиме доступа для записи и возвращает полученный из неё
141  → результат.</para>
142  /// </summary>
143  /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
144  → результата функции.</para></typeparam>
145  /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
146  → параметра.</para></typeparam>
147  /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
148  → параметра.</para></typeparam>
149  /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
150  → параметра.</para></typeparam>
151  /// <param name="synchronization"><para>Synchronization
152  → object.</para><para>Синхронизация объекта.</para></param>
153  /// <param name="parameter1"><para>The first parameter</para><para>Первый
154  → параметр.</para></param>
155  /// <param name="parameter2"><para>The second parameter</para><para>Второй
156  → параметр.</para></param>
157  /// <param name="parameter3"><para>The third parameter</para><para>Третий
158  → параметр.</para></param>
159  /// <param name="function"><para>The function.</para><para>Функция.</para></param>
160  /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
161  public static TResult ExecuteWriteOperation<TResult, TParam1, TParam2, TParam3>(this
162  → ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2, TParam3
163  → parameter3, Func<TParam1, TParam2, TParam3, TResult> function) =>
164  → synchronization.ExecuteWriteOperation(() => function(parameter1, parameter2,
165  → parameter3));
166
167  /// <summary>
168  /// <para>Executes action in write access mode.</para>
169  /// <para>Выполняет действие в режиме доступа для записи.</para>
170  /// </summary>
171  /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
172  → параметра.</para></typeparam>
173  /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
174  → параметра.</para></typeparam>
175  /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
176  → параметра.</para></typeparam>
177  /// <param name="synchronization"><para>Synchronization
178  → object.</para><para>Синхронизация объекта.</para></param>
179  /// <param name="parameter1"><para>The first parameter</para><para>Первый
180  → параметр.</para></param>
181  /// <param name="parameter2"><para>The second parameter</para><para>Второй
182  → параметр.</para></param>
183  /// <param name="parameter3"><para>The third parameter</para><para>Третий
184  → параметр.</para></param>
185  /// <param name="action"><para>The action.</para><para>Действие.</para></param>
186  public static void ExecuteWriteOperation<TParam1, TParam2, TParam3>(this
187  → ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2, TParam3
188  → parameter3, Action<TParam1, TParam2, TParam3> action) =>
189  → synchronization.ExecuteWriteOperation(() => action(parameter1, parameter2,
190  → parameter3));
191
192  /// <summary>
193  /// <para>Executes a function in read access mode and returns the function's
194  → result.</para>
195  /// <para>Выполняет функцию в режиме доступа для чтения и возвращает полученный из неё
196  → результат.</para>
197  /// </summary>
198  /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
199  → результата функции.</para></typeparam>
200  /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
201  → параметра.</para></typeparam>
202  /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
203  → параметра.</para></typeparam>
204  /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
205  → параметра.</para></typeparam>
206  /// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
207  → параметра.</para></typeparam>
208  /// <param name="synchronization"><para>Synchronization
209  → object.</para><para>Синхронизация объекта.</para></param>
210  /// <param name="parameter1"><para>The first parameter</para><para>Первый
211  → параметр.</para></param>

```

```

178  /// <param name="parameter2"><para>The second parameter</para><para>Второй
    ↳ параметр.</para></param>
179  /// <param name="parameter3"><para>The third parameter</para><para>Третий
    ↳ параметр.</para></param>
180  /// <param name="parameter4"><para>The forth parameter</para><para>Чертвёртый
    ↳ параметр.</para></param>
181  /// <param name="function"><para>The function.</para><para>Функция.</para></param>
182  /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
183  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));

184  /// <summary>
185  /// <para>Executes action in read access mode.</para>
186  /// <para>Выполняет действие в режиме доступа для чтения.</para>
187  /// </summary>
188  /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
    ↳ параметра.</para></typeparam>
189  /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
    ↳ параметра.</para></typeparam>
190  /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
    ↳ параметра.</para></typeparam>
191  /// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
    ↳ параметра.</para></typeparam>
192  /// <param name="synchronization"><para>Synchronization
    ↳ object.</para><para>Синхронизация объекта.</para></param>
193  /// <param name="parameter1"><para>The first parameter</para><para>Первый
    ↳ параметр.</para></param>
194  /// <param name="parameter2"><para>The second parameter</para><para>Второй
    ↳ параметр.</para></param>
195  /// <param name="parameter3"><para>The third parameter</para><para>Третий
    ↳ параметр.</para></param>
196  /// <param name="parameter4"><para>The forth parameter</para><para>Чертвёртый
    ↳ параметр.</para></param>
197  /// <param name="action"><para>The action.</para><para>Действие.</para></param>
198  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));

200  /// <summary>
201  /// <para>Executes a function in write access mode and returns the function's
    ↳ result.</para>
202  /// <para>Выполняет функцию в режиме доступа для записи и возвращает полученный из неё
    ↳ результат.</para>
203  /// </summary>
204  /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
    ↳ результата функции.</para></typeparam>
205  /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
    ↳ параметра.</para></typeparam>
206  /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
    ↳ параметра.</para></typeparam>
207  /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
    ↳ параметра.</para></typeparam>
208  /// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
    ↳ параметра.</para></typeparam>
209  /// <param name="synchronization"><para>Synchronization
    ↳ object.</para><para>Синхронизация объекта.</para></param>
210  /// <param name="parameter1"><para>The first parameter</para><para>Первый
    ↳ параметр.</para></param>
211  /// <param name="parameter2"><para>The second parameter</para><para>Второй
    ↳ параметр.</para></param>
212  /// <param name="parameter3"><para>The third parameter</para><para>Третий
    ↳ параметр.</para></param>
213  /// <param name="parameter4"><para>The forth parameter</para><para>Чертвёртый
    ↳ параметр.</para></param>
214  /// <param name="function"><para>The function.</para><para>Функция.</para></param>
215  /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
216

```

```

217 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));
218
219 /// <summary>
220 /// <para>Executes action in write access mode.</para>
221 /// <para>Выполняет действие в режиме доступа для записи.</para>
222 /// </summary>
223 /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
    ↳ параметра.</para></typeparam>
224 /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
    ↳ параметра.</para></typeparam>
225 /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
    ↳ параметра.</para></typeparam>
226 /// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
    ↳ параметра.</para></typeparam>
227 /// <param name="synchronization"><para>Synchronization
    ↳ object.</para><para>Синхронизация объекта.</para></param>
228 /// <param name="parameter1"><para>The first parameter</para><para>Первый
    ↳ параметр.</para></param>
229 /// <param name="parameter2"><para>The second parameter</para><para>Второй
    ↳ параметр.</para></param>
230 /// <param name="parameter3"><para>The third parameter</para><para>Третий
    ↳ параметр.</para></param>
231 /// <param name="parameter4"><para>The forth parameter</para><para>Четвёртый
    ↳ параметр.</para></param>
232 /// <param name="action"><para>The action.</para><para>Действие.</para></param>
233 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));
234 }
235 }

```

./Synchronization/ISynchronized.cs

```

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 synchronization 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
            ↳ guarantee 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
            ↳ guarantee thread safe access synchronization.</para>
28         /// <para>Возвращает обернутую/декорированную версию <typeparamref name="TInterface"/>,
            ↳ которая гарантирует потокобезопасную синхронизацию доступа.</para>
29         /// </summary>
30         /// <remarks>

```

```

31     /// <para>It is safe to use it directly, because it must be thread safe
    ↪ implementation.</para>
32     /// <para>Безопасно использовать напрямую, так как реализация должна быть
    ↪ потокобезопасной.</para>
33     /// </remarks>
34     TInterface Sync { get; }
35 }
36 }

```

./Synchronization/ReaderWriterLockSynchronization.cs

```

1  using System;
2  using System.Threading;
3
4  namespace Platform.Threading.Synchronization
5  {
6      /// <summary>
7      /// <para>Implementation of <see cref="ISynchronization"/> based on <see
    ↪ cref="ReaderWriterLockSlim"/>.</para>
8      /// <para>Реализация <see cref="ISynchronization"/> на основе <see
    ↪ cref="ReaderWriterLockSlim"/>.</para>
9      /// </summary>
10     public class ReaderWriterLockSynchronization : ISynchronization
11     {
12         private readonly ReaderWriterLockSlim _rwLock = new
    ↪ ReaderWriterLockSlim(LockRecursionPolicy.SupportsRecursion);
13
14         /// <inheritdoc />
15         /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
    ↪ path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
    ↪ n.ExecuteReadOperation(System.Action)"]/*' />
16         public void ExecuteReadOperation(Action action)
17         {
18             _rwLock.EnterReadLock();
19             try
20             {
21                 action();
22             }
23             finally
24             {
25                 _rwLock.ExitReadLock();
26             }
27         }
28
29         /// <inheritdoc />
30         /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
    ↪ path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
    ↪ n.ExecuteReadOperation`1(System.Func{`0})"]/*' />
31         public TResult ExecuteReadOperation<TResult>(Func<TResult> function)
32         {
33             _rwLock.EnterReadLock();
34             try
35             {
36                 return function();
37             }
38             finally
39             {
40                 _rwLock.ExitReadLock();
41             }
42         }
43
44         /// <inheritdoc />
45         /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
    ↪ path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
    ↪ n.ExecuteWriteOperation(System.Action)"]/*' />
46         public void ExecuteWriteOperation(Action action)
47         {
48             _rwLock.EnterWriteLock();
49             try
50             {
51                 action();
52             }
53             finally
54             {
55                 _rwLock.ExitWriteLock();
56             }
57         }
58
59         /// <inheritdoc />

```



```

60     /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
        ↳ path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
        ↳ n.ExecuteWriteOperation`1(System.Func{`0})"]/*' />
61 public TResult ExecuteWriteOperation<TResult>(Func<TResult> function)
62 {
63     _rwLock.EnterWriteLock();
64     try
65     {
66         return function();
67     }
68     finally
69     {
70         _rwLock.ExitWriteLock();
71     }
72 }
73 }
74 }

```

./Synchronization/Unsynchynchronization.cs

```

1  using System;
2
3  namespace Platform.Threading.Synchronization
4  {
5      /// <summary>
6      /// <para>Implementation of <see cref="ISynchronization"/> that makes no actual
        ↳ synchronization.</para>
7      /// <para>Реализация <see cref="ISynchronization"/>, которая не выполняет фактическую
        ↳ синхронизацию.</para>
8      /// </summary>
9      public class Unsynchynchronization : ISynchronization
10     {
11         /// <inheritdoc />
12         /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
            ↳ path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↳ n.ExecuteReadOperation(System.Action)"]/*' />
13         public void ExecuteReadOperation(Action action) => action();
14
15         /// <inheritdoc />
16         /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
            ↳ path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↳ n.ExecuteReadOperation`1(System.Func{`0})"]/*' />
17         public TResult ExecuteReadOperation<TResult>(Func<TResult> function) => function();
18
19         /// <inheritdoc />
20         /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
            ↳ path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↳ n.ExecuteWriteOperation(System.Action)"]/*' />
21         public void ExecuteWriteOperation(Action action) => action();
22
23         /// <inheritdoc />
24         /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
            ↳ path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↳ n.ExecuteWriteOperation`1(System.Func{`0})"]/*' />
25         public TResult ExecuteWriteOperation<TResult>(Func<TResult> function) => function();
26     }
27 }

```

./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) =>
15             ↳ SyncInvokeWithExtendedStack(param, action, ExtendedMaxStackSize);
16
17         public static void SyncInvokeWithExtendedStack<T>(T param, Action<object> action, int
18             ↳ maxStackSize) => StartNew(param, action, maxStackSize).Join();
19
20         public static void SyncInvokeWithExtendedStack(Action action) =>
21             ↳ SyncInvokeWithExtendedStack(action, ExtendedMaxStackSize);
22
23         public static void SyncInvokeWithExtendedStack(Action action, int maxStackSize) =>
24             ↳ StartNew(action, maxStackSize).Join();
25
26         public static Thread StartNew<T>(T param, Action<object> action) => StartNew(param,
27             ↳ action, DefaultMaxStackSize);
28
29         public static Thread StartNew<T>(T param, Action<object> action, int maxStackSize)
30         {
31             var thread = new Thread(new ParameterizedThreadStart(action), maxStackSize);
32             thread.Start(param);
33             return thread;
34         }
35
36         public static Thread StartNew(Action action) => StartNew(action, DefaultMaxStackSize);
37
38         public static Thread StartNew(Action action, int maxStackSize)
39         {
40             var thread = new Thread(new ThreadStart(action), maxStackSize);
41             thread.Start();
42             return thread;
43         }
44
45         public static void Sleep() => Thread.Sleep(DefaultSleepTimeout);
46     }
47 }

```

Index

- ./ConcurrentQueueExtensions.cs, 1
- ./Synchronization/ISynchronization.cs, 1
- ./Synchronization/ISynchronizationExtensions.cs, 2
- ./Synchronization/ISynchronized.cs, 7
- ./Synchronization/ReaderWriterLockSynchronization.cs, 8
- ./Synchronization/Unsynchynchronization.cs, 9
- ./TaskExtensions.cs, 9
- ./ThreadHelpers.cs, 9