

RedisWorkQueue

Generated by Doxygen 1.9.7

1 WorkQueue	1
1 WorkQueue	1
1.0.1 Properties	1
1.0.2 Methods	2
1.0.3 Example Usage	2
2 Namespace Documentation	3
2.1 RedisWorkQueue Namespace Reference	3
3 Class Documentation	3
3.1 RedisWorkQueue.Item Class Reference	3
3.1.1 Detailed Description	3
3.1.2 Constructor & Destructor Documentation	3
3.1.3 Member Function Documentation	4
3.1.4 Property Documentation	5
3.2 RedisWorkQueue.KeyPrefix Class Reference	5
3.2.1 Detailed Description	5
3.2.2 Constructor & Destructor Documentation	5
3.2.3 Member Function Documentation	6
3.2.4 Property Documentation	6
4 File Documentation	7
4.1 Item.cs	7
4.2 KeyPrefix.cs	7
4.3 WorkQueue.cs	8
Index	11

1 WorkQueue

The `WorkQueue` class represents a work queue backed by a Redis database. It provides methods to add items to the queue, lease items from the queue, and mark completed items as done.

1.0.1 Properties

- `Session`: Gets or sets the unique identifier for the current session.
- `MainQueueKey`: Gets or sets the Redis key for the main queue.
- `ProcessingKey`: Gets or sets the Redis key for the processing queue.
- `CleaningKey`: Gets or sets the Redis key for the cleaning queue.
- `LeaseKey`: Gets or sets the Redis key prefix for lease keys.

1.0.2 Methods

- `AddItem(IRedisClient db, Item item)`: Adds an item to the work queue. The `db` parameter is the Redis instance and the `item` parameter is the item to be added.
- `QueueLength(IRedisClient db)`: Gets the length of the main queue. The `db` parameter is the Redis instance.
- `Processing(IRedisClient db)`: Gets the length of the processing queue. The `db` parameter is the Redis instance.
- `LeaseExists(IRedisClient db, string itemId)`: Checks if a lease exists for the specified item ID. The `db` parameter is the Redis instance and the `itemId` parameter is the ID of the item to check.
- `Lease(IRedisClient db, int leaseSeconds, bool block, int timeout = 0)`↵
: Requests a work lease from the work queue. This should be called by a worker to get work to complete. The `db` parameter is the Redis instance, the `leaseSeconds` parameter is the number of seconds to lease the item for, the `block` parameter indicates whether to block and wait for an item to be available if the main queue is empty, and the `timeout` parameter is the maximum time to block in milliseconds.
- `Complete(IRedisClient db, Item item)`: Marks a job as completed and removes it from the work queue. The `db` parameter is the Redis instance and the `item` parameter is the item to be completed.

1.0.3 Example Usage

```
using FreeRedis;
using RedisWorkQueue;

var redis = new RedisClient("localhost");
var workQueue = new WorkQueue("work_queue");

var item = new Item(Encoding.UTF8.GetBytes("data"), "item_1");

workQueue.AddItem(redis, item);

var queueLength = workQueue.QueueLength(redis);
Console.WriteLine($"Queue Length: {queueLength}");

var lease = workQueue.Lease(redis, 30, true, 10000);
if (lease != null)
{
    Console.WriteLine($"Leased Item: {lease.ID}");
    // Do the work here
    workQueue.Complete(redis, lease);
}
else
{
    Console.WriteLine("No item available to lease");
}
```

In this example, we create a Redis client and a new instance of the `WorkQueue` class. We then add an item to the work queue using the `AddItem` method and get the length of the main queue using the `QueueLength` method.

We then try to lease an item from the work queue using the `Lease` method. If an item is available, we do the work and mark the item as completed using the `Complete` method.

Note that in this example, we pass `true` for the `block` parameter of the `Lease` method, which means the method will block and wait for an item to be available if the main queue is empty. We also pass a `timeout` value of 10000 milliseconds, which means that if there are no items available after 10 seconds, the method will return `null`.

2 Namespace Documentation

2.1 RedisWorkQueue Namespace Reference

Classes

- class [Item](#)
Represents an item to be stored in the Redis work queue.
- class [KeyPrefix](#)
KeyPrefix is a string which should be prefixed to an identifier to generate a database key.

3 Class Documentation

3.1 RedisWorkQueue.Item Class Reference

Represents an item to be stored in the Redis work queue.

Public Member Functions

- [Item](#) (object [Data](#), string? [ID](#)=null)
Creates a new instance of the Item class with the specified data and ID.
- T? [DataJson](#)< T > ()
Deserializes the stored data into an object using JSON deserialization.

Static Public Member Functions

- static [Item FromJson](#) (object data, string? id=null)
Creates a new instance of the Item class from the provided data by serializing it as JSON.

Properties

- byte[] [Data](#) [get, set]
Gets or sets the serialized data as a byte array.
- string [ID](#) [get, set]
Gets or sets the ID of the item.

3.1.1 Detailed Description

Represents an item to be stored in the Redis work queue.

Definition at line 12 of file [Item.cs](#).

3.1.2 Constructor & Destructor Documentation

Item()

```
RedisWorkQueue.Item.Item (
    object Data,
    string? ID = null ) [inline]
```

Creates a new instance of the Item class with the specified data and ID.

Parameters

<i>data</i>	The data to be serialized and stored in the item.
<i>id</i>	An optional ID to uniquely identify the item. If not provided, a new GUID will be generated.

Gets or sets the serialized data as a byte array.

Definition at line 29 of file [Item.cs](#).

3.1.3 Member Function Documentation**DataJson< T >()**

```
T? RedisWorkQueue.Item.DataJson< T > ( ) [inline]
```

Deserializes the stored data into an object using JSON deserialization.

Template Parameters

<i>T</i>	The type to deserialize the data into.
----------	--

Returns

The deserialized object of type T. Returns null if the deserialization fails.

Definition at line 79 of file [Item.cs](#).

FromJson()

```
static Item RedisWorkQueue.Item.FromJson (
    object data,
    string? id = null ) [inline], [static]
```

Creates a new instance of the Item class from the provided data by serializing it as JSON.

Parameters

<i>data</i>	The data to be serialized and stored in the item.
<i>id</i>	An optional ID to identify the item. If not provided, a new GUID will be generated.

Returns

A new instance of the Item class with the serialized JSON data.

Definition at line 69 of file [Item.cs](#).

3.1.4 Property Documentation

Data

```
byte [] RedisWorkQueue.Item.Data [get], [set]
```

Gets or sets the serialized data as a byte array.

Definition at line 17 of file [Item.cs](#).

ID

```
string RedisWorkQueue.Item.ID [get], [set]
```

Gets or sets the ID of the item.

Definition at line 22 of file [Item.cs](#).

The documentation for this class was generated from the following file:

- [RedisWorkQueue/Item.cs](#)

3.2 RedisWorkQueue.KeyPrefix Class Reference

KeyPrefix is a string which should be prefixed to an identifier to generate a database key.

Public Member Functions

- [KeyPrefix](#) (string [Prefix](#))
Creates a new instance of the KeyPrefix class with the specified prefix.

Static Public Member Functions

- static [KeyPrefix Concat](#) ([KeyPrefix](#) prefix, string name)
Concat other onto prefix and return the result as a KeyPrefix.

Properties

- string [Prefix](#) [get, set]
Gets or sets the prefix string.

3.2.1 Detailed Description

KeyPrefix is a string which should be prefixed to an identifier to generate a database key.

Definition at line 6 of file [KeyPrefix.cs](#).

3.2.2 Constructor & Destructor Documentation

KeyPrefix()

```
RedisWorkQueue.KeyPrefix.KeyPrefix (  
    string Prefix ) [inline]
```

Creates a new instance of the KeyPrefix class with the specified prefix.

Parameters

<i>prefix</i>	A string specifying the prefix to use for Redis keys.
---------------	---

Definition at line 17 of file [KeyPrefix.cs](#).

3.2.3 Member Function Documentation**Concat()**

```
static KeyPrefix RedisWorkQueue.KeyPrefix.Concat (
    KeyPrefix prefix,
    string name ) [inline], [static]
```

Concat other onto prefix and return the result as a KeyPrefix.

Parameters

<i>prefix</i>	An instance of the KeyPrefix class representing the prefix to concatenate.
<i>name</i>	Name to concatenate with the prefix.

Returns

A new KeyPrefix instance with the concatenated namespaced prefix.

Definition at line 38 of file [KeyPrefix.cs](#).

3.2.4 Property Documentation**Prefix**

```
string RedisWorkQueue.KeyPrefix.Prefix [get], [set]
```

Gets or sets the prefix string.

Definition at line 11 of file [KeyPrefix.cs](#).

The documentation for this class was generated from the following file:

- RedisWorkQueue/KeyPrefix.cs

4 File Documentation

4.1 Item.cs

```

00001 using System;
00002 using System.IO;
00003 using System.Runtime.Serialization.Formatters.Binary;
00004 using System.Text;
00005 using Newtonsoft.Json;
00006
00007 namespace RedisWorkQueue
00008 {
00012     public class Item
00013     {
00017         public byte[] Data { get; set; }
00018
00022         public string ID { get; set; }
00023
00029         public Item(object Data, string? ID = null)
00030         {
00034             byte[] byteData;
00035             if (Data is string)
00036                 byteData = Encoding.UTF8.GetBytes((string)Data);
00037             else if (!(Data is byte[]))
00038             {
00039                 BinaryFormatter bf = new BinaryFormatter();
00040                 using (var ms = new MemoryStream())
00041                 {
00042                     //as long as we have full control over and know what the data is then this is okay
00043                     #pragma warning disable SYSLIB0011
00044                     bf.Serialize(ms, Data);
00045                     #pragma warning restore SYSLIB0011
00046                     byteData = ms.ToArray();
00047                 }
00048             }
00049             else
00050                 byteData = (byte[])Data;
00051
00052             if (ID == null) ID = Guid.NewGuid().ToString();
00053
00054             if (byteData == null)
00055                 throw new Exception("item failed to serialise data to byte[]");
00056             this.Data = byteData;
00057             if (ID == null)
00058                 throw new Exception("item failed to create ID");
00059
00060             this.ID = ID;
00061         }
00062
00069         public static Item FromJson(object data, string? id = null)
00070         {
00071             return new Item(JsonConvert.SerializeObject(data), id);
00072         }
00073
00079         public T? DataJson<T>()
00080         {
00081             return JsonConvert.DeserializeObject<T>(Encoding.UTF8.GetString(Data));
00082         }
00083     }
00084 }

```

4.2 KeyPrefix.cs

```

00001 namespace RedisWorkQueue
00002 {
00006     public class KeyPrefix
00007     {
00011         public string Prefix { get; set; }
00012
00017         public KeyPrefix(string Prefix)
00018         {
00019             this.Prefix = Prefix;
00020         }
00021
00022
00028         {
00029             return Prefix + name;
00030         }
00031
00038         public static KeyPrefix Concat(KeyPrefix prefix, string name)
00039         {

```



```

00040         return new KeyPrefix(prefix.Of(name));
00041     }
00042 }
00043 }

```

4.3 WorkQueue.cs

```

00001 using System.Text;
00002 using FreeRedis;
00003
00004 namespace RedisWorkQueue
00005 {
00009     public class WorkQueue
00010     {
00014         public class WorkQueue
00015         {
00019             public string Session { get; set; }
00020
00024             public string MainQueueKey { get; set; }
00025
00029             public string ProcessingKey { get; set; }
00030
00034             public string CleaningKey { get; set; }
00035
00039             public KeyPrefix LeaseKey { get; set; }
00040
00041
00046             public WorkQueue(KeyPrefix name)
00047             {
00048                 this.Session = name.Of(Guid.NewGuid().ToString());
00049                 this.MainQueueKey = name.Of(":queue");
00050                 this.ProcessingKey = name.Of(":processing");
00051                 this.CleaningKey = name.Of(":cleaning");
00052                 this.LeaseKey = KeyPrefix.Concat(name, ":leased_by_session:");
00053                 this.ItemDataKey = KeyPrefix.Concat(name, ":item:");
00054             }
00055
00061             public void AddItem(IRedisClient db, Item item)
00062             {
00063                 using (var pipe = db.StartPipe())
00064                 {
00065                     pipe.Set(ItemDataKey.Of(item.ID), item.Data);
00066                     pipe.LPush(MainQueueKey, item.ID);
00067
00068                     pipe.EndPipe();
00069                 }
00070             }
00071
00077             public long QueueLength(IRedisClient db)
00078             {
00079                 return db.LLen(MainQueueKey);
00080             }
00081
00087             public long Processing(IRedisClient db)
00088             {
00089                 return db.LLen(ProcessingKey);
00090             }
00091
00098             public bool LeaseExists(IRedisClient db, string itemId)
00099             {
00100                 return db.Exists(LeaseKey.Of(itemId));
00101             }
00102
00103
00118             public Item? Lease(IRedisClient db, int leaseSeconds, bool block, int timeout = 0)
00119             {
00120                 object maybeItemId;
00121                 if (block)
00122                 {
00123                     maybeItemId = db.BRPopLPush(MainQueueKey, ProcessingKey, timeout);
00124                 }
00125                 else
00126                 {
00127                     maybeItemId = db.RPopLPush(MainQueueKey, ProcessingKey);
00128                 }
00129
00130                 if (maybeItemId == null)
00131                     return null;
00132
00133                 string itemId;
00134                 if (maybeItemId is byte[])
00135                     itemId = Encoding.UTF8.GetString((byte[])maybeItemId);
00136                 else if (maybeItemId is string)
00137                     itemId = (string)maybeItemId;

```

```
00138         else
00139             throw new Exception("item id from work queue not bytes or string");
00140
00141         var data = db.Get<byte[]>(ItemDataKey.Of(itemId));
00142         if (data == null)
00143             data = new byte[0];
00144
00145         db.SetEx(LeaseKey.Of(itemId), leaseSeconds, Encoding.UTF8.GetBytes(Session));
00146
00147         return new Item(data, itemId);
00148     }
00149
00150     public bool Complete(IRedisClient db, Item item)
00151     {
00152         var removed = db.LRem(ProcessingKey, 0, item.ID);
00153
00154         if (removed == 0)
00155             return false;
00156
00157         string itemId = item.ID;
00158
00159         using (var pipe = db.StartPipe())
00160         {
00161             pipe.Del(ItemDataKey.Of(itemId));
00162             pipe.Del(LeaseKey.Of(itemId));
00163
00164             pipe.EndPipe();
00165         }
00166
00167         return true;
00168     }
00169 }
00170 }
```


Index

Concat

RedisWorkQueue.KeyPrefix, [6](#)

Data

RedisWorkQueue.Item, [5](#)

DataJson< T >

RedisWorkQueue.Item, [4](#)

FromJson

RedisWorkQueue.Item, [4](#)

ID

RedisWorkQueue.Item, [5](#)

Item

RedisWorkQueue.Item, [3](#)

KeyPrefix

RedisWorkQueue.KeyPrefix, [5](#)

Prefix

RedisWorkQueue.KeyPrefix, [6](#)

RedisWorkQueue, [3](#)

RedisWorkQueue.Item, [3](#)

Data, [5](#)

DataJson< T >, [4](#)

FromJson, [4](#)

ID, [5](#)

Item, [3](#)

RedisWorkQueue.KeyPrefix, [5](#)

Concat, [6](#)

KeyPrefix, [5](#)

Prefix, [6](#)

RedisWorkQueue/Item.cs, [7](#)

RedisWorkQueue/KeyPrefix.cs, [7](#)

RedisWorkQueue/WorkQueue.cs, [8](#)

WorkQueue, [1](#)