

 免費電子書

學習

# StackExchange.Redis

Free unaffiliated eBook created from  
**Stack Overflow contributors.**

#stackexch

ange.redis

.....	1
<b>1: StackExchange.Redis</b> .....	<b>2</b>
.....	2
.....	2
.....	2
.....	2
Examples.....	2
.....	2
.....	2
.....	2
<b>2:</b> .....	<b>4</b>
.....	4
Examples.....	4
.....	4
.....	5
<b>3:</b> .....	<b>6</b>
.....	6
.....	6
.....	6
Examples.....	6
.....	6
.....	6
<b>4:</b> .....	<b>7</b>
Examples.....	7
.....	7
<b>5:</b> .....	<b>8</b>
Examples.....	8
.....	8
JSON.....	8
Protobuf.....	9
<b>6:</b> .....	<b>11</b>

Examples.....	11
.....	11
int.....	11
.....	<b>12</b>

---

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [stackexchange-redis](#)

It is an unofficial and free StackExchange.Redis ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official StackExchange.Redis.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to [info@zzzprojects.com](mailto:info@zzzprojects.com)

# 1: StackExchange.Redis

[Stack.change.RedisNuget](#) [Github](#) ◦

•

```
1.0.187 2014318
```

## Examples

```
using StackExchange.Redis;

// ...

// connect to the server
ConnectionMultiplexer connection = ConnectionMultiplexer.Connect("localhost");

// select a database (by default, DB = 0)
IDatabase db = connection.GetDatabase();

// run a command, in this case a GET
RedisValue myVal = db.StringGet("mykey");
```

```
class Program
{
    private static Lazy<ConnectionMultiplexer> _multiplexer =
        new Lazy<ConnectionMultiplexer>(
            () => ConnectionMultiplexer.Connect("localhost"),
            LazyThreadSafetyMode.ExecutionAndPublication);

    static void Main(string[] args)
    {
        IDatabase db1 = _multiplexer.Value.GetDatabase(1);
        IDatabase db2 = _multiplexer.Value.GetDatabase(2);
    }
}
```

## Redisadminrisky

```
ConfigurationOptions options = new ConfigurationOptions()
{
    EndPoints = { { "localhost", 6379} },
    AllowAdmin = true,
    ConnectTimeout = 60*1000,
};
ConnectionMultiplexer multiplexer = ConnectionMultiplexer.Connect(options);
```

```
ConnectionMultiplexer multiplexer =
    ConnectionMultiplexer.Connect("localhost:6379,allowAdmin=True,connectTimeout=60000");
```

## SSLRedis

```
ConfigurationOptions options = new ConfigurationOptions()
    {
        EndPoints = { { "localhost", 6380}},
        Ssl = true,
        Password = "12345"
    };
ConnectionMultiplexer multiplexer = ConnectionMultiplexer.Connect(options);
```

```
ConnectionMultiplexer multiplexer =
    ConnectionMultiplexer.Connect("localhost:6380,ssl=True,password=12345");
```

[StackExchange.Redis](https://riptutorial.com/zh-TW/stackexchange-redis/topic/1/stackexchange-redis) <https://riptutorial.com/zh-TW/stackexchange-redis/topic/1/stackexchange-redis>

## 2:

**StackExchange.Redis** `IProfilerConnectionMultiplexer.RegisterProfiler(IProfiler)`

`ConnectionMultiplexer.BeginProfiling(object)` `ConnectionMultiplexer.FinishProfiling(object)`

`object`

`IProfiler` `BeginStackExchange.RedisFinish`

## Examples

```
class ToyProfiler : IProfiler
{
    public ConcurrentDictionary<Thread, object> Contexts = new ConcurrentDictionary<Thread, object>();

    public object GetContext()
    {
        object ctx;
        if(!Contexts.TryGetValue(Thread.CurrentThread, out ctx)) ctx = null;

        return ctx;
    }
}

// ...

ConnectionMultiplexer conn = /* initialization */;
var profiler = new ToyProfiler();
var thisGroupContext = new object();

conn.RegisterProfiler(profiler);

var threads = new List<Thread>();

for (var i = 0; i < 16; i++)
{
    var db = conn.GetDatabase(i);

    var thread =
        new Thread(
            delegate()
            {
                var threadTasks = new List<Task>();

                for (var j = 0; j < 1000; j++)
                {
                    var task = db.StringSetAsync("" + j, "" + j);
                    threadTasks.Add(task);
                }

                Task.WaitAll(threadTasks.ToArray());
            }
        );
}
```

```

    profiler.Contexts[thread] = thisGroupContext;

    threads.Add(thread);
}

conn.BeginProfiling(thisGroupContext);

threads.ForEach(thread => thread.Start());
threads.ForEach(thread => thread.Join());

IEnumerable<IProfiledCommand> timings = conn.FinishProfiling(thisGroupContext);

```

## 16,000IProfiledCommand - redis。

```

ConnectionMultiplexer conn = /* initialization */;
var profiler = new ToyProfiler();

conn.RegisterProfiler(profiler);

var threads = new List<Thread>();

var perThreadTimings = new ConcurrentDictionary<Thread, List<IProfiledCommand>>();

for (var i = 0; i < 16; i++)
{
    var db = conn.GetDatabase(i);

    var thread =
        new Thread(
            delegate()
            {
                var threadTasks = new List<Task>();

                conn.BeginProfiling(Thread.CurrentThread);

                for (var j = 0; j < 1000; j++)
                {
                    var task = db.StringSetAsync("" + j, "" + j);
                    threadTasks.Add(task);
                }

                Task.WaitAll(threadTasks.ToArray());

                perThreadTimings[Thread.CurrentThread] =
conn.FinishProfiling(Thread.CurrentThread).ToList();
            }
        );

    profiler.Contexts[thread] = thread;

    threads.Add(thread);
}

threads.ForEach(thread => thread.Start());
threads.ForEach(thread => thread.Join());

```

perThreadTimings 161000IProfilingCommandsThread。

<https://riptutorial.com/zh-TW/stackexchange-redis/topic/4/>



### 3:

- `public IEnumerable<RedisKey> Keys(int database = 0, RedisValue pattern = default(RedisValue), int pageSize = CursorUtils.DefaultPageSize, long cursor = CursorUtils.Origin, int pageOffset = 0, CommandFlags flags = CommandFlags.None)`

	Redis
pageOffset	

`Keys()` **Redis** `KEYSSCAN` `SCAN` `IEnumerable<RedisKey>` `KEYS`

## Examples

```
// Connect to a target server using your ConnectionMultiplexer instance
IServer server = conn.GetServer("localhost", 6379);

// Write out each key in the server
foreach(var key in server.Keys()) {
    Console.WriteLine(key);
}
```

```
// Connect to a target server using your ConnectionMultiplexer instance
IServer server = conn.GetServer("localhost", 6379);

var seq = server.Keys();
IScanningCursor scanningCursor = (IScanningCursor)seq;

// Use the cursor in some way...
```

<https://riptutorial.com/zh-TW/stackexchange-redis/topic/66/>

## 4:

# Examples

```
var multiplexer = ConnectionMultiplexer.Connect("localhost");
IDatabase db = multiplexer.GetDatabase();

// initialize key with empty string
await db.StringSetAsync("key", "");

// create transaction that utilize multiplexing and pipelining
ITransaction transacton = db.CreateTransaction();
Task<long> appendA = transacton.StringAppendAsync("key", "a");
Task<long> appendB = transacton.StringAppendAsync("key", "b");

if (await transacton.ExecuteAsync()) // sends "MULTI APPEND KEY a APPEND KEY b EXEC
// in single request to redis server
{
    // order here doesn't matter, result is always - "abc".
    // 'a' and 'b' append always together in isolation of other Redis commands
    // 'c' appends to "ab" because transaction is already executed successfully
    await appendA;
    await db.StringAppendAsync("key", "c");
    await appendB;
}

string value = db.StringGet("key"); // value is "abc"
```

<https://riptutorial.com/zh-TW/stackexchange-redis/topic/3217/>

# 5:

## Examples

`ISubscriber.Publish`

```
// grab an instance of an ISubscriber
var subscriber = connection.GetSubscriber();

// publish a message to the 'chat' channel
subscriber.Publish("chat", "This is a message")
```

`ISubscriber.Subscribe` ◦ ◦

```
// grab an instance of an ISubscriber
var subscriber = connection.GetSubscriber();

// subscribe to a messages over the 'chat' channel
subscriber.Subscribe("chat", (channel, message) => {
    // do something with the message
    Console.WriteLine((string)message);
});
```

## JSON

```
// definition of a message
public class ChatMessage
{
    public Guid Id { get; set; }
    public string User { get; set; }
    public string Text { get; set; }
}

// grab an instance of an ISubscriber
var subscriber = connection.GetSubscriber();

var message = new ChatMessage
{
    Id = Guid.NewGuid(),
    User = "User 1234",
    Text = "Hello World!"
};

// serialize a ChatMessage
// this uses JIL to serialize to JSON
var json = JSON.Serialize(message);

// publish the message to the 'chat' channel
subscriber.Publish("chat", json)
```

```
// grab an instance of an ISubscriber
var subscriber = connection.GetSubscriber();
```

```
// subscribe to messages over the 'chat' channel
subscriber.Subscribe("chat", (channel, json) => {
    var message = JSON.Deserialize<ChatMessage>(json);

    // do something with the message
    Console.WriteLine($"{message.User} said {message.Text}");
});
```

## Protobuf

### StackExchange.Redispub / subprotobuf-net

```
// definition of a message (marked up with Protobuf attributes)
[ProtoContract]
public class ChatMessage
{
    [ProtoMember(1)]
    public Guid Id { get; set; }
    [ProtoMember(2)]
    public string User { get; set; }
    [ProtoMember(3)]
    public string Text { get; set; }
}

// grab an instance of an ISubscriber
var subscriber = connection.GetSubscriber();

var message = new ChatMessage
{
    Id = Guid.NewGuid(),
    User = "User 1234",
    Text = "Hello World!"
};

using (var memoryStream = new MemoryStream())
{
    // serialize a ChatMessage using protobuf-net
    Serializer.Serialize(memoryStream, message);

    // publish the message to the 'chat' channel
    subscriber.Publish("chat", memoryStream.ToArray());
}
```

```
// grab an instance of an ISubscriber
var subscriber = connection.GetSubscriber();

// subscribe to messages over the 'chat' channel
subscriber.Subscribe("chat", (channel, bytes) => {
    using (var memoryStream = new MemoryStream(bytes))
    {
        var message = Serializer.Deserialize<ChatMessage>(memoryStream);

        // do something with the message
        Console.WriteLine($"{message.User} said {message.Text}");
    }
});
```

<https://riptutorial.com/zh-TW/stackexchange-redis/topic/1610/>

---

# 6:

## Examples

### RedisRedisValue

```
/"myvalue" here is implicitly converted to a RedisValue type  
//The RedisValue type is rarely seen in practice.  
db.StringSet("key", "aValue");
```

### int

```
db.StringSet("key", 11021);  
int i = (int)db.StringGet("key");
```

### [StackExchange.Redis.Extensions](#)

```
db.Add("key", 11021);  
int i = db.Get<int>("key");
```

<https://riptutorial.com/zh-TW/stackexchange-redis/topic/11/>

---

S. No		Contributors
1	StackExchange.Redis	<a href="#">Adam Lear</a> , <a href="#">Community</a> , <a href="#">Kevin Montrose</a> , <a href="#">Nilay Vishwakarma</a> , <a href="#">Vladimir Dorokhov</a>
2		<a href="#">Jason Punyon</a> , <a href="#">Kevin Montrose</a> , <a href="#">Shog9</a>
3		<a href="#">Joseph Vaughan</a>
4		<a href="#">Vladimir Dorokhov</a>
5		<a href="#">Dean Ward</a>
6		<a href="#">Arie Litovsky</a> , <a href="#">Cigano Morrison Mendez</a> , <a href="#">Kevin Montrose</a>